周回積分に基づく固有値解法の大規模計算環境における密行列向け並列実装に関する研究

著者	矢野 貴大
発行年	2019
学位授与大学	筑波大学 (University of Tsukuba)
学位授与年度	2018
報告番号	12102甲第8993号
URL	http://doi.org/10.15068/00156281

周回積分に基づく固有値解法の
大規模計算環境における密行列向け
並列実装に関する研究

２０１９年　３月

矢野 貴大

周回積分に基づく固有値解法の
大規模計算環境における密行列向け
並列実装に関する研究

矢野 貴大

システム情報工学研究科
筑波大学

２０１９年　３月

論文要旨

本論文では，大規模計算環境における密行列向け一般化固有値問題の内部固有値問題を対象とした高性能・高並列性を併せ持つ固有値ソルバの実現を目標としている．本論文で扱う周回積分型固有値解法は従来大規模疎行列向け解法として研究されており，計算機クラスタにおいて高い並列性能が示されている．同法を高性能な密行列向け線形計算ライブラリを用いて実装することで，密行列向けの固有値解法として高性能・高並列性の実現を目指す．

周回積分型固有値解法では解法の内部で線形ソルバを使用している．大規模密行列向け分散並列線形ソルバとして block Krylov 部分空間反復法を密行列向けに実装することによって，高性能な分散並列線形ソルバを実現する．さらに，特定の性質を持つ連立一次方程式を複素数演算無しで解く real-valued 解法を提案し，実対称定値一般化固有値問題向け解法の高速化を図る．

目次

図目次

表目次

アルゴリズム目次

第 1 章

序論

構造設計や流体解析，物性解析など様々な科学・工学の分野で数値シミュレーションが行われている．数値シミュレーションの計算において固有値問題という問題が現れ，その求解にかかる時間が計算時間の大部分を占めることが多い．並列計算環境の大規模・高性能化に伴って数値シミュレーションの規模も大規模化しており，ボトルネックになりやすい固有値問題の求解を高速化することは科学技術計算における重要な課題である．

固有値問題とは，行列値関数 $F : \Omega \subset \mathbb{C} \to \mathbb{C}^{n\times n}$ について

$$F(\lambda)\boldsymbol{x} = \mathbf{0} \tag{1.1}$$

を満たす固有値 $\lambda \in \Omega$ および対応する固有ベクトル $\boldsymbol{x} \in \mathbb{C}^n \backslash \{\mathbf{0}\}$ を求める問題である．特に $F(\lambda)$ が

$$F(\lambda) = \lambda I - A, \quad A \in \mathbb{C}^{n\times n}, \tag{1.2}$$

$$F(\lambda) = \lambda B - A, \quad A, B \in \mathbb{C}^{n\times n}, \tag{1.3}$$

の形である場合をそれぞれ標準固有値問題および一般化固有値問題と呼び，$F(\lambda)$ が λ に対して非線形の項を含む場合を非線形固有値問題と呼ぶ．また，固有値および対応する固有ベクトルまとめて固有対と呼ぶ．固有値問題の中でも，全ての固有対ではなく一部の固有対を求めるような問題を部分固有値問題と呼ぶ．

本論文では一般化固有値問題を取り扱う．一般化固有値問題において，A, B が共に実対称行列であり B が定値である場合を実対称定値一般化固有値問題と呼び，A, B が共にエルミート行列で B が定値である場合をエルミート定値一般化固有値問題と呼ぶ．また，これらをまとめて定値一般化固有値問題と呼ぶ．定値一般化固有値問題は振動解析や分子軌道計算，電子状態計算など科学技術計算の数値シミュレーションにおいて広く現れる問題である．

定値一般化固有値問題を解く方法として，相似変換を用いた手法 [25] が研究されており広く使用されている．この手法は定値一般化固有値問題に対する直接法と称されることがある．解法は 1)B の Cholesky 分解を利用して標準固有値問題へ変換，2) 変換された標準固有値問題を構成する行列を相似変換を用いて三重対角 (またはより広い幅の帯) 行列へ変換，3) 三重対角 (または帯) 行列の固有値および対応する固有ベクトルの求解，4) 固有ベクトルの逆変換，の大まかに 4 つの手順によって構成

されている．この解法の分散並列実装では，2) および 4) のステップにおいて相似変換で生じる通信がボトルネックになりやすく，高い並列性能を発揮することは困難である．また計算機のアーキテクチャが複雑化しており，各環境への移植および性能のチューニングは困難である．

近年の大規模計算環境の計算ノードでは，CPU 以外の計算資源として GPU などのコプロセッサを搭載することが一般的となってきている．CPU についても，マルチコア CPU や，Intel Xeon Phi などのメニーコアプロセッサを搭載することが一般的となってきており，メニーコアを活用して計算の高速化を行うようなアルゴリズムやその実装が重要である．ノード間のインターコネクトについてもバンド幅が広いものやレイテンシが低いものなど様々なものがあり，大規模計算環境を最大限活用するためには，インターコネクトの性能も考慮した上でアルゴリズムやその実装を開発する必要がある．

近年，大規模並列計算環境に適した固有値解法として周回積分型固有値解法 [30] が提案されている．周回積分型固有値解法は周回積分により指定の領域内に含まれる固有ベクトルのみを含む部分空間を生成し，その部分空間から固有対を抽出する射影法である．射影法は主に大規模疎行列向けの解法として研究されている手法である．周回積分型固有値解法についても主に大規模疎行列向けの解法として研究されており，密行列への適用はあまり研究されていない．周回積分型固有値解法として Sakurai-Sugiura 法 [30]，FEAST algorithm[29]，そして Beyn の方法 [3] が挙げられる．

計算機上では周回積分を厳密に計算することが困難であることから，数値積分によって近似的に計算を行う．数値積分の計算において，複数の独立した連立一次方程式の求解が必要であり，周回積分型固有値解法の計算時間の大部分はこの連立一次方程式の求解にかかる時間である．このため，内部で使用する線形ソルバ高速化することは固有値解法としての高速化に直結する．

GPU を搭載した計算環境において，GPU を活用して線形計算を行う計算ライブラリとして cuBLAS[5] や MAGMA[24, 33] が挙げられる．特に MAGMA はノード内に搭載されている複数の GPU を使用して計算を行うことが可能である．MAGMA には LU 分解を複数 GPU で行う関数が実装されていることから，それを利用して周回積分型固有値解法を実装することで GPU クラスタにおける密行列向けの高性能・高スケーラビリティな密行列向け一般化固有値解法の分散並列実装を実現することができると考えられる．

MAGMA はノード内の GPU を用いた並列計算は可能であるが，ノード間の並列計算には対応していない．そのため，MAGMA において扱うことができる問題のサイズはノードに搭載されている GPU の基数および GPU のデバイスメモリの容量に律せられる．大規模な行列の連立一次方程式を解くためには複数ノードによる分散並列な線形ソルバが必要となる．しかし，GPU を使用し複数ノードによる分散並列実行可能な並列線形ソルバは現時点では困難である．GPU や Xeon Phi などのメニーコア環境においては行列-行列積 (GEMM) を高速に計算することが可能である．そのため，block Krylov 部分空間反復法を密行列向けに実装することで，計算の大部分が GEMM になることから，高性能な線形ソルバを開発できるのではないかと考えられる．

一般化固有値問題の中でも，実対称一般化固有値問題は応用上一番良く現れる問題である．そのため，実対称一般化固有値問題を周回積分型固有値解法を用いて解く際に現れる，連立一次方程式を効率良く解くことができる手法の開発が望まれる．

本研究では大規模メニーコア環境における高性能な分散並列固有値ソルバの実現を目的とする．そのため，本論文では中規模密行列向け固有値ソルバとして周回積分型固有値解法を MPI および線形計算ライブラリ MAMGA および cuBLAS を用いた実装を提案する．MAGMA の直接法線形ソルバに対応できない大きさの問題に対応するために，メニーコア環境を活かした分散並列線形ソルバとして，block GEMM ベースの block Krylov 部分空間反復法の分散並列実装を提案する．また，応用上重要となる実対称定値一般化固有値問題を効率よく解くための新しい block Krylov 部分空間反復法も提案する．

1.1 本論文の構成

本論文の構成を述べる．第 2 章では，相似変換型固有値解法と周回積分型固有値解法について述べる．第 3 章では，GPU クラスタにおける周回積分型固有値解法の中規模密行列向け分散並列実装手法を提案し，性能の評価を行う．第 4 章では，前章の手法を大規模密行列に対して適用する際に問題となる連立一次方程式解法について分散並列実装手法を提案し，既存の手法と性能の比較を行う．第 5 章では，実対称定値一般化固有値問題に対して Sakurai-Sugiura 法を適用する際に現れる連立一次方程式を効率的に解く手法を提案し，既存の手法と比較を行う．最後に第 6 章で本論文の結論を述べる．

第 2 章

相似変換型固有値解法と周回積分型固有値解法

本章では先行研究である定値一般化固有値問題向け相似変換型解法，および一般化固有値問題向け周回積分型固有値解法について述べる．

2.1　相似変換型固有値解法

エルミート定値一般化固有値問題

$$A\boldsymbol{x} = \lambda B\boldsymbol{x}, \quad \lambda \in \mathbb{R}, \quad \boldsymbol{x} \in \mathbb{C}^n \backslash \{\boldsymbol{0}\} \tag{2.1}$$

を考える．ここで，$A, B \in \mathbb{C}^{n\times n}$ は共にエルミート行列であり，B は正定値である．

B がエルミート正定値であることから，B の Cholesky 分解

$$B = LL^{\mathrm{H}} \tag{2.2}$$

を考える．

$$\boldsymbol{y} := L^{\mathrm{H}}\boldsymbol{x} \tag{2.3}$$

と定義することで元の一般化固有値問題 (2.1) は

$$\tilde{A}\boldsymbol{y} = \lambda\boldsymbol{y} \tag{2.4}$$

と標準固有値問題へ変換される．ここで，

$$\tilde{A} := L^{-1}AL^{-\mathrm{H}} \tag{2.5}$$

である．$\tilde{A}$ に対して Householder 変換を使用して上 Hessenberg 行列に変換することを考える．このとき，$\tilde{A}$ はエルミート行列であることから，Householder 変換によるユニタリ行列 $Q \in \mathbb{C}^{n\times n}$ を用いて

$$T = Q^{\mathrm{H}}\tilde{A}Q \tag{2.6}$$

と実対称三重対角行列 $T \in \mathbb{R}^{n\times n}$ へ変換する事が可能である．

$$\boldsymbol{z} := Q^{\mathrm{H}}\boldsymbol{y} \tag{2.7}$$

を利用すると，標準固有値問題 (2.4) は

$$T\boldsymbol{z} = \lambda\boldsymbol{z} \tag{2.8}$$

と実対称三重対角行列の標準固有値問題へ更に変換される．相似変換では固有値が変化しないことから，T の固有値を求めることは $\tilde{A}$ の固有値を求めることと等価であり，元の一般化固有値問題 (2.1) の固有値を求めることと等価である．何らかの方法で標準固有値問題 (2.8) の固有対 $(\lambda, \boldsymbol{z})$ を計算した上で，(2.7) および (2.3) の関係を用いて固有ベクトルを逆変換することにより元の一般化固有値問題 (2.1) の固有対を計算する．

実三重対角行列の固有対を求める手法として，全固有値を計算する必要がある場合は QR 法 [15] および分割統治法に基づく方法 [6] が，部分固有値が必要である場合には，二分法 [34] により固有値を求めた上で各固有値に対して逆反復法 [35] により固有ベクトルを計算する手法や，Multiple Relatively Robust Representations (MRRR) に基づく方法 [9] が挙げられる．

2.2 周回積分型固有値解法

本節では周回積分型固有値解法である Sakurai-Sugiura 法について，理論および特徴について述べる．

2.2.1 周回積分による部分空間の生成と Rayleigh-Ritz procedure

求める領域を $\Omega \subset \mathbb{C}$ とし，Ω に沿った正の向きの周回路を Γ とする．L 本の線形独立な列ベクトルからなる行列 $V \in \mathbb{C}^{n\times L}$ に対して Γ 上の周回積分により行列 S を

$$\begin{aligned} S &:= [S_0, \ldots, S_{M-1}] \in \mathbb{C}^{n\times LM}, \\ S_k &:= \frac{1}{2\pi\mathrm{i}} \oint_{\Gamma} z^k \left(zB - A\right)^{-1} BV \mathrm{d}z, \quad k = 0, \ldots, M-1, \end{aligned} \tag{2.9}$$

と定義する．ここで，i は虚数単位である．L をブロックサイズと呼び，Ω 内に含まれる固有値の最大の重複度以上とする．また，M をモーメント次数と呼び，LM が Ω 内の重複を含めた固有値の数 m 以上とする．このとき，S の列ベクトルは Ω 内の固有値に対応する固有ベクトルの線形結合となる．つまり，

$$\mathrm{span}(S) = \mathrm{span}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m).$$

行列 S から固有値および対応する固有ベクトルを抽出するための手法として，block Hankel 行列を用いた手法 [19]，Rayleigh-Ritz procedure による手法 [18] そして block Arnoldi 法に基づく手法 [20] が挙げられる．本研究では，Rayleigh-Ritz procedure による手法を利用する．S の特異値分解を

$$S = U\Sigma W^{\mathrm{H}}$$

とする．ここで，$\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_{LM}) \in \mathbb{R}^{LM \times LM}$，$U = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{LM}] \in \mathbb{C}^{n \times LM}$，$W \in \mathbb{C}^{LM \times LM}$ である．rank $S = m$ であるから，特異値について，

$$\sigma_1 \geq \cdots \geq \sigma_m > \sigma_{m+1} = \cdots = \sigma_{LM} = 0, \tag{2.10}$$

の関係が成立する．非零特異値に対応する左特異ベクトルからなる行列 U_m を

$$U_m = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m],$$

と定める．U_m を用いて小規模な一般化固有値問題

$$U_m^{\mathrm{H}} A U_m \boldsymbol{r} = \omega U_m^{\mathrm{H}} B U_m \boldsymbol{r}$$

を解き，固有値 ω および対応する固有ベクトル $\boldsymbol{r}$ を得る．$\lambda = \omega$ および $\boldsymbol{x} = U_m \boldsymbol{r}$ とすることで，元の固有値問題の固有値および対応する固有ベクトルを得られる．

2.2.2 数値積分による近似

周回積分 (2.9) を計算機上で厳密に計算することは困難であるため，N 個の積分点を用いた数値積分により近似計算を行う．領域 Ω が中心 $\gamma \in \mathbb{C}$，実軸方向の半径 $\rho \in \mathbb{R}$ および実軸方向の半径と虚軸方向の半径の比 $\alpha \in \mathbb{R}$ で表現される楕円であるとする．このとき，S_k は

$$S_k \approx \hat{S}_k = \rho \sum_{j=1}^{N} w_j \zeta_j^k \left(z_j B - A\right)^{-1} BV, \tag{2.11}$$

と近似計算される．ここで，ζ_j は正規化された積分点，z_j は実際の積分点，w_j は z_j における数値積分の重み，θ_j は領域が正円である場合の z_j が位置する角度であり，それぞれ

$$\begin{aligned} w_j &= \alpha \cos\theta_j + \mathrm{i} \sin\theta_j, \\ \zeta_j &= \cos\theta_j + \mathrm{i}\alpha \sin\theta_j, \\ z_j &= \gamma + \rho\zeta_j, \\ \theta_j &= \frac{2\pi}{N}\left(j - \frac{1}{2}\right), \end{aligned}$$

である．以降，数値積分による誤差を含む量は $\hat{}$ を付けて表記する．

$\hat{S}$ の特異値分解を

$$\hat{S} = \hat{U}\hat{\Sigma}\hat{W}^{\mathrm{H}}$$

とする．$\hat{S}$ は一般に rank $\hat{S} \neq m$ であるため，関係 (2.10) が成立しない．このため，小さなしきい値 $\delta \in \mathbb{R}$ を用いて

$$\hat{\sigma}_1 \geq \cdots \geq \hat{\sigma}_{\hat{m}} \geq \delta\hat{\sigma}_1 > \hat{\sigma}_{\hat{m}+1} \geq \cdots \geq 0, \tag{2.12}$$

を満たす $\hat{m}$ を $\hat{S}$ の数値ランクとする．U_m の代わりに $\hat{U}_{\hat{m}}$ を用いることで元の問題を小規模な固有値問題

$$\hat{U}_{\hat{m}}^{\mathrm{H}} A \hat{U}_{\hat{m}} \hat{\boldsymbol{r}} = \hat{\omega} \hat{U}_{\hat{m}}^{\mathrm{H}} B \hat{U}_{\hat{m}} \hat{\boldsymbol{r}}$$

Algorithm 1 Sakurai-Sugiura method with Rayleigh-Ritz procedure

Input: $A, B \in \mathbb{C}^{n\times n},\ L, M, N \in \mathbb{N},\ \alpha, \gamma, \delta \in \mathbb{R},\ \rho \in \mathbb{C}$

Output: $\lambda_i \in \mathbb{C}, \boldsymbol{x}_i \in \mathbb{C}^n, i = 1, \ldots, \hat{m}$

1: Generate an input matrix V
2: **for** $j = 1, \ldots, N$ **do**
3: Calculate $w_j, \zeta_j, z_j, \theta_j$
4: Solve $(z_j B - A)^{-1} Y_j = BV$
5: **end for**
6: Calculate $\hat{S}$ by (2.11)
7: Perform singular value decomposition of $\hat{S}$
8: Determine $\hat{m}$ by (2.12)
9: Form the projected eigenproblem
10: Solve the projected eigenproblem
11: Calculate $(\lambda_i, \boldsymbol{x}_i) = (\omega_i, \hat{U}_{\hat{m}} \boldsymbol{r}_i)$

に帰着させ，$\hat{\omega}$ および $\hat{\boldsymbol{r}}$ を得る．$\lambda = \hat{\omega}$ および $\boldsymbol{x} = \hat{U}_{\hat{m}} \hat{\boldsymbol{r}}$ とすることで，元の固有値問題の固有値および対応する固有ベクトルを近似的に計算することが可能である．以上をまとめると，**Algorithm 1** となる．

2.2.3 階層的並列性

Sakurai-Sugiura 法は領域を複数置くことができ，各々の領域における計算は全て独立して行うことが可能である．これを上層 (Top Layer) の並列性と呼ぶ．各領域内の計算において，N 個の連立一次方程式の解を計算する必要があるが，各々の連立一次方程式の係数行列は独立であるため同時に計算を行うことが可能である．これを中層 (Middle Layer) の並列性と呼ぶ．また各連立一次方程式の解の計算に並列な連立一次方程式解法を採用することができる．これを下層 (Bottom Layer) の並列性と呼ぶ．このように Sakurai-Sugiura 法は階層的並列性を有しており，各層の並列性を計算機クラスタに割り当てることで計算機クラスタの性能を十分に活用することができる固有値解法となっている．図 2.1 に階層的並列性の概念図を示す．

2.2.4 問題の対称性を用いた計算量削減

Sakurai-Sugiura 法の計算において，最もコストがかかる計算は (2.11) における N 個の L 本の右辺ベクトルを持つ連立一次方程式

$$(z_j B - A) Y_j = BV,\ Y_j \in \mathbb{C}^{n\times L}, \quad j = 1, \ldots, N, \tag{2.13}$$

の求解である．

行列 A および B が共に実対称行列またはエルミート行列の場合を考える．この場合，固有値は全

て実軸上に位置することから領域 Ω を実軸に対象となるように設置する．N が偶数であるとき，積分点 z_j について，

$$z_j = \overline{z_{N-j+1}}, \tag{2.14}$$

の関係が成立する．

行列 A および B が実対称行列の場合，各積分点における連立一次方程式 (2.13) の係数行列は複素対称行列となり，入力行列 V を実行列であると仮定すると，

$$C_j = \overline{C_{N-j+1}},$$
$$Y_j = \overline{Y_{N-j+1}},$$

の関係が成立するので，式 (2.11) は，

$$\hat{S}_k = 2\rho \sum_{j=1}^{N/2} \mathrm{Re}\left(w_j \zeta_j^k Y_j\right),$$

となる．連立一次方程式を直接法線形ソルバを用いて解く場合，分解および前進後退代入はそれぞれ $N/2$ 回に削減される．

行列 A および B がエルミート行列である場合，各積分点での連立一次方程式 (2.13) の係数行列は，

$$C_j = C_{N-j+1}^{\mathrm{H}}, \tag{2.15}$$

の関係が成立する．このため直接法線形ソルバを用いる場合，行列分解の回数を $N/2$ 回に削減することができる．

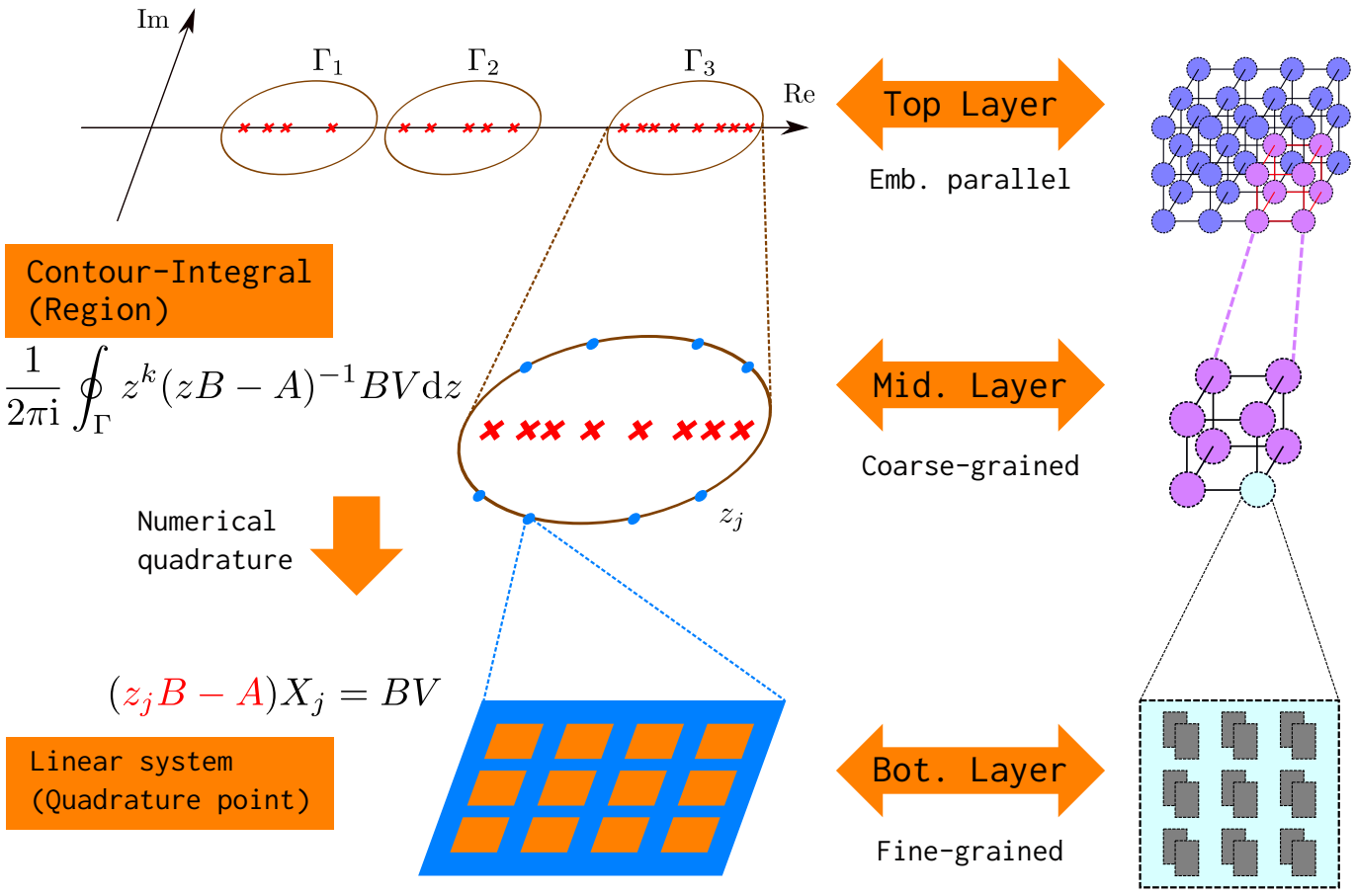

図 2.1　Sakurai-Sugiura 法の階層的並列性の概念図

第 3 章

中規模密行列向け Sakurai-Sugiura 法の分散並列実装

前章では Sakurai-Sugiura 法のアルゴリズムについて述べた．同法は前章で述べた通り階層的並列性を持つ．この階層的並列性を活用するために，通信ライブラリ MPI を用いた分散並列実装手法が考えられる．

本章では中規模密行列からなる一般化固有値問題向け Sakurai-Sugiura 法の GPU クラスタ向け分散並列実装，および計算時間モデルを提案する．一領域を対象とし，複数領域を設置して計算する場合は設置する領域の数だけコミュニケータを分割し，各コミュニケータに対して領域を割り当てる．以下，ある領域について計算するコミュニケータのプロセス数を N_p とする．また，1 ノードあたり 1 プロセスが割り当てられ，各プロセスはノード内の全ての GPU にアクセスできることを仮定する．GPU 上における線形計算は，線形計算ライブラリ MAGMA および cuBLAS によって行われる．

3.1 $\hat{S}$ の計算

コミュニケータに属する N_p プロセスに対して，積分点 N 個を割り当てる．各プロセスは高々 $\lceil \frac{N}{N_p} \rceil$ 個の積分点を担当する．ここで，$\lceil x \rceil$ は x を下回らない最小の整数を示す．プロセス $i(= 0, \ldots, N_p - 1)$ が担当する積分点の添字の集合を $\mathcal{N}^{(i)}$ とする．各計算ノードにおいて連立一次方程式の求解は MAGMA の `magma_zgetrf_mgpu`[23] を用いてノード内にある全ての GPU を用いて LU 分解を行い，LAPACK の `ZGETRS` 関数を用いて前進後退代入を行う．`magma_zgetrf_mgpu` はあらかじめ行列データをデバイスメモリに格納しておく必要があるのでホストからデバイスへ行列データの通信を行う．LU 分解の計算結果はデバイスメモリ上にある係数行列に上書きされる形で計算される．`ZGETRS` はホストで計算を行うので，計算結果をホストに戻すための通信が必要となる．次に連立一次方程式の解 Y_j を用いて各プロセスは $\hat{S}$ の一部の計算を行う．これらの計算を割り当てられ

た積分点の数だけ繰り返し行う．この結果プロセス i は

$$\hat{S}^{(i)} = \begin{bmatrix} \hat{S}_0^{(i)} & \cdots & \hat{S}_{M-1}^{(0)} \end{bmatrix} \in \mathbb{C}^{n \times LM},$$
$$\hat{S}_k^{(i)} = \rho \sum_{j \in \mathcal{N}^{(i)}} w_j \zeta_j^k Y_j, \quad k = 0, \ldots, M-1,$$

を保持する．最後にコミュニケータ上で $\hat{S}^{(i)}$ に対して `MPI_Allreduce` を使用して足し合わせることで $\hat{S}$ を計算しつつ全プロセスに配布する．

3.2 Rayleigh-Ritz procedure

$\hat{S}$ について直交基底の生成および数値ランクの決定のために $\hat{S}$ の特異値分解を行う．このとき，$\hat{S}$ の特異値分解を直接計算するのではなく一度

$$\hat{S} = QR$$

と QR 分解を計算した後，R に対して特異値分解

$$R = U \Sigma W^{\mathrm{H}}$$

を行い，最後に

$$\hat{U}_{\hat{m}} = Q \begin{bmatrix} \boldsymbol{u}_1 & \cdots & \boldsymbol{u}_{\hat{m}} \end{bmatrix}$$

として計算を行う．この計算は各プロセスににおいて冗長に計算を行う．

直交基底を用いて元の問題の行列 A および行列 B の射影行列を計算する．メモリ使用量削減のため行列 A および行列 B を行方向に $s = \max(N, p)$ 分割し，コミュニケータ上のプロセスに計算を担当する添字を割り当てる．

$$A = \begin{bmatrix} A_1 \\ \vdots \\ A_s \end{bmatrix}, \quad B = \begin{bmatrix} B_1 \\ \vdots \\ B_s \end{bmatrix}.$$

割り当てられた添字の集合を $\mathcal{J}$ とする．$j \in \mathcal{J}$ について，$\hat{U}_{\hat{m}}^{\mathrm{H}} A_j \hat{U}_{\hat{m}}$ および $\hat{U}_{\hat{m}}^{\mathrm{H}} B_j \hat{U}_{\hat{m}}$ を計算し，割り当てられた全ての添字に対して計算を行った後，計算結果を `MPI_Allreduce` を用いて収集することにより $\hat{U}_{\hat{m}}^{\mathrm{H}} A \hat{U}_{\hat{m}}$ および $\hat{U}_{\hat{m}}^{\mathrm{H}} B \hat{U}_{\hat{m}}$ を計算する．

小さな一般化固有値問題および固有ベクトルの逆変換について，MAGMA または LAPACK の関数を用いてコミュニケータ上の全プロセスが冗長に計算を行う．

以上をまとめると，**Algorithm 2** となる．また，問題の対称性を利用した場合の実装として，実対称定値一般化固有値問題向け実装を **Algorithm 3** に，エルミート定値一般化固有値問題向け実装を **Algorithm 4** にそれぞれ示す．

3.3 計算時間モデル

前節で提案した並列実装の性能を計算時間モデルで表現する．

Algorithm 2 Parallel SSM implementation for medium sized dense matrices on GPU clusters

Input: $A, B \in \mathbb{C}^{n\times n},\ L, M, N \in \mathbb{N},\ \alpha, \gamma, \delta \in \mathbb{R},\ \rho \in \mathbb{C}$

Output: $\lambda_i \in \mathbb{C}, \boldsymbol{x}_i \in \mathbb{C}^n, i = 1, \ldots, \hat{m}$

1: Generate an input matrix V
2: Compute BV `cublas_zgemm`
3: Broadcast BV `MPI_Bcast`
4: Set $\hat{S}_k^{(i)} := 0$
5: **for** $j \in \mathcal{N}^{(i)}$ **do**
6: Compute $C_j = z_j B - A$
7: Transfer C_j to device(s)
8: Factorize $C_j = L_j U_j$ `magma_zgetrf_mgpu`
9: Transfer LU factors from device(s)
10: Solve $C_j Y_j = BV$ `ZGETRS`
11: **for** $k = 0, \ldots, M-1$ **do**
12: $\hat{S}_k^{(i)} := \hat{S}_k^{(i)} + \rho w_j \zeta_j^k Y_j$
13: **end for**
14: **end for**
15: Allreduce $\hat{S}^{(i)}$ with summation `MPI_Allreduce`
16: Perform QR decomposition: $\hat{S} = QR$ `magma_zgeqrf`
17: Perform singular value decomposition: $R = \hat{U}'\hat{\Sigma}\hat{W}^{\mathrm{H}}$ `magma_zgesvd`
18: Determine $\hat{m}$
19: Compute $\hat{U}_{\hat{m}} = Q\,[\hat{\boldsymbol{u}}_1, \ldots, \hat{\boldsymbol{u}}_{\hat{m}}]$ `magma_zunmqr`
20: **for** $j \in \mathcal{J}$ **do**
21: Compute $A_j \hat{U}_{\hat{m}}$ `cublas_zgemm`
22: Compute $\hat{U}_{\hat{m}}^{\mathrm{H}} A_j \hat{U}_{\hat{m}}$ `cublas_zgemm`
23: Compute $B_j \hat{U}_{\hat{m}}$ `cublas_zgemm`
24: Compute $\hat{U}_{\hat{m}}^{\mathrm{H}} B_j \hat{U}_{\hat{m}}$ `cublas_zgemm`
25: **end for**
26: Allreduce $\hat{U}_{\hat{m}}^{\mathrm{H}} A_j \hat{U}_{\hat{m}}$ and $\hat{U}_{\hat{m}}^{\mathrm{H}} B_j \hat{U}_{\hat{m}}$ with summation `MPI_Allreduce`
27: Solve $\hat{U}_{\hat{m}}^{\mathrm{H}} A_j \hat{U}_{\hat{m}} \boldsymbol{t}_i = \theta_i \hat{U}_{\hat{m}}^{\mathrm{H}} B_j \hat{U}_{\hat{m}} \boldsymbol{t}_i,\ i = 1, \ldots, \hat{m}$ `ZGGEV`
28: Compute $\boldsymbol{x}_i = \hat{U}_{\hat{m}} \boldsymbol{t}_i, i = 1,\ \ldots, \hat{m}$ `cublas_zgemm`
29: $\lambda_i = \theta_i, i = 1,\ \ldots, \hat{m}$

Algorithm 3 Specialized version for SSPD type problems

Input: $A, B \in \mathbb{R}^{n\times n},\ L, M, N \in \mathbb{N},\ \alpha, \gamma, \delta, \rho \in \mathbb{R}$

Output: $\lambda_i \in \mathbb{R}, \boldsymbol{x}_i \in \mathbb{R}^n, i = 1, \ldots, \hat{m}$

1:	Generate an input matrix V	
2:	Compute BV	`cublas_dgemm`
3:	Broadcast BV	`MPI_Bcast`
4:	Set $\hat{S}_k^{(i)} := 0$	
5:	**for** $j \in \mathcal{N}^{(i)}$ **do**	
6:	Compute $C_j = z_j B - A$	
7:	Transfer C_j to device(s)	
8:	Factorize $C_j = L_j U_j$	`magma_zgetrf_mgpu`
9:	Transfer LU factors from device(s)	
10:	Solve $C_j Y_j = BV$	`ZGETRS`
11:	**for** $k = 0, \ldots, M-1$ **do**	
12:	$\hat{S}_k^{(i)} := \hat{S}_k^{(i)} + 2\rho \mathrm{Re}\left(w_j \zeta_j^k Y_j\right)$	
13:	**end for**	
14:	**end for**	
15:	Allreduce $\hat{S}^{(i)}$ with summation	`MPI_Allreduce`
16:	Perform QR decomposition: $\hat{S} = QR$	`magma_dgeqrf`
17:	Perform singular value decomposition: $R = \hat{U}'\hat{\Sigma}\hat{W}^{\mathrm{H}}$	`magma_dgesvd`
18:	Determine $\hat{m}$	
19:	Compute $\hat{U}_{\hat{m}} = Q\left[\hat{\boldsymbol{u}}_1, \ldots, \hat{\boldsymbol{u}}_{\hat{m}}\right]$	`magma_dormqr`
20:	**for** $j \in \mathcal{J}$ **do**	
21:	Compute $A_j \hat{U}_{\hat{m}}$	`cublas_dgemm`
22:	Compute $\hat{U}_{\hat{m}}^{\mathrm{H}} A_j \hat{U}_{\hat{m}}$	`cublas_dgemm`
23:	Compute $B_j \hat{U}_{\hat{m}}$	`cublas_dgemm`
24:	Compute $\hat{U}_{\hat{m}}^{\mathrm{H}} B_j \hat{U}_{\hat{m}}$	`cublas_dgemm`
25:	**end for**	
26:	Allreduce $\hat{U}_{\hat{m}}^{\mathrm{H}} A_j \hat{U}_{\hat{m}}$ and $\hat{U}_{\hat{m}}^{\mathrm{H}} B_j \hat{U}_{\hat{m}}$ with summation	`MPI_Allreduce`
27:	Solve $\hat{U}_{\hat{m}}^{\mathrm{H}} A_j \hat{U}_{\hat{m}} \boldsymbol{t}_i = \theta_i \hat{U}_{\hat{m}}^{\mathrm{H}} B_j \hat{U}_{\hat{m}} \boldsymbol{t}_i,\ i = 1, \ldots, \hat{m}$	`magma_dgesvd`
28:	Compute $\boldsymbol{x}_i = \hat{U}_{\hat{m}} \boldsymbol{t}_i,\ i = 1, \ldots, \hat{m}$	`cublas_dgemm`
29:	$\lambda_i = \theta_i,\ i = 1, \ldots, \hat{m}$	

Algorithm 4 Specialized version for HHPD type problems

Input: $A, B \in \mathbb{C}^{n\times n},\ L, M, N \in \mathbb{N},\ \alpha, \gamma, \delta, \rho \in \mathbb{R}$

Output: $\lambda_i \in \mathbb{R}, \boldsymbol{x}_i \in \mathbb{C}^n, i = 1, \ldots, \hat{m}$

1:	Generate an input matrix V	
2:	Compute BV	`cublas_*gemm`
3:	Broadcast BV	`MPI_Bcast`
4:	Set $\hat{S}_k^{(i)} := 0$	
5:	**for** $j \in \mathcal{N}^{(i)}$ **do**	
6:	Compute $C_j = z_j B - A$	
7:	Transfer C_j to device(s)	
8:	Factorize $C_j = L_j U_j$	`magma_zgetrf_mgpu`
9:	Transfer LU factors from device(s)	
10:	Solve $C_j Y_j = BV$	`ZGETRS`
11:	Solve $C_j^{\mathrm{H}} Y_{N-j+1} = BV$	`ZGETRS`
12:	**for** $k = 0, \ldots, M-1$ **do**	
13:	$\hat{S}_k^{(i)} := \hat{S}_k^{(i)} + \rho w_j \zeta_j^k Y_j$	
14:	$\hat{S}_k^{(i)} := \hat{S}_k^{(i)} + \rho w_{N-j+1} \zeta_{N-j+1}^k Y_{N-j+1}$	
15:	**end for**	
16:	**end for**	
17:	Allreduce $\hat{S}^{(i)}$ with summation	`MPI_Allreduce`
18:	Perform QR decomposition: $\hat{S} = QR$	`magma_zgeqrf`
19:	Perform singular value decomposition: $R = \hat{U}' \hat{\Sigma} \hat{W}^{\mathrm{H}}$	`magma_zgesvd`
20:	Determine $\hat{m}$	
21:	Compute $\hat{U}_{\hat{m}} = Q\,[\hat{\boldsymbol{u}}_1, \ldots, \hat{\boldsymbol{u}}_{\hat{m}}]$	`magma_zunmqr`
22:	**for** $j \in \mathcal{J}$ **do**	
23:	Compute $A_j \hat{U}_{\hat{m}}$	`cublas_zgemm`
24:	Compute $\hat{U}_{\hat{m}}^{\mathrm{H}} A_j \hat{U}_{\hat{m}}$	`cublas_zgemm`
25:	Compute $B_j \hat{U}_{\hat{m}}$	`cublas_zgemm`
26:	Compute $\hat{U}_{\hat{m}}^{\mathrm{H}} B_j \hat{U}_{\hat{m}}$	`cublas_zgemm`
27:	**end for**	
28:	Allreduce $\hat{U}_{\hat{m}}^{\mathrm{H}} A_j \hat{U}_{\hat{m}}$ and $\hat{U}_{\hat{m}}^{\mathrm{H}} B_j \hat{U}_{\hat{m}}$ with summation	`MPI_Allreduce`
29:	Solve $\hat{U}_{\hat{m}}^{\mathrm{H}} A_j \hat{U}_{\hat{m}} \boldsymbol{t}_i = \theta_i \hat{U}_{\hat{m}}^{\mathrm{H}} B_j \hat{U}_{\hat{m}} \boldsymbol{t}_i,\ i = 1, \ldots, \hat{m}$	`magma_zhegvd`
30:	Compute $\boldsymbol{x}_i = \hat{U}_{\hat{m}} \boldsymbol{t}_i,\ i = 1, \ldots, \hat{m}$	`cublas_zgemm`
31:	$\lambda_i = \theta_i,\ i = 1, \ldots, \hat{m}$	

表 3.1　HA-PACS ベースクラスタ部の計算ノードの仕様

CPU	Intel Xeon E5-2670, 2.60GHz, 2 socket, 332.8GFLOPS/node
Memory	128GB, DDR3 1600MHz, 102.8GB/s/node
GPU	NVIDIA M2090, 4 GPU, 2660GFLOPS/node
GPU memory	6GB/GPU, GDDR5 177GB/s (ECC off)
Interconnect	Infiniband QDR × 2 レーン

表 3.2　使用したソフトウェア

コンパイラ	GCC 4.4.7
MPI	MVAPICH2 1.8.1
CUDA	5.5.22
Intel MKL	11.1.0
MAGMA	1.5.0 beta2

1 プロセスでの計算時間 $T_{\mathrm{total}}(1)$ は次式で表される．

$$T_{\mathrm{total}}(1) = N\{T_{\mathrm{fact}}(n_{\mathrm{GPU}}) + T_{\mathrm{solve}}\} + T_{\mathrm{SVD}} + T_{\mathrm{RR}}. \tag{3.1}$$

ここで，$T_{\mathrm{fact}}(n_{\mathrm{GPU}})$ は n_{GPU}GPU を使用して LU 分解を計算する時間，T_{solve} は全身後退代入の計算時間，T_{SVD} は特異値分解を計算する時間，そして T_{RR} は $\hat{U}_{\hat{m}}$ による Rayleigh-Ritz procedure の計算時間である．

次に，N_p プロセスを積分点の並列性に対して適用した場合を考える．N 点分の LU 分解および全身後退代入が N_p プロセスに割り当てられるので，計算時間 $T_{\mathrm{total}}(N_p)$ は次式で表される．

$$T_{\mathrm{total}}(N_p) = \left\lceil \frac{N}{N_p} \right\rceil \{T_{\mathrm{fact}}(n_{\mathrm{GPU}}) + T_{\mathrm{solve}}\} + T_{\mathrm{SVD}} + T_{\mathrm{RR}}. \tag{3.2}$$

特に，$N_p = N$ であれば，

$$T_{\mathrm{total}}(N_p) = T_{\mathrm{fact}}(n_{\mathrm{GPU}}) + T_{\mathrm{solve}} + T_{\mathrm{SVD}} + T_{\mathrm{RR}}, \tag{3.3}$$

となる．

3.4　数値実験

提案した実装の性能を評価するために，数値実験を行った．実験は，筑波大学計算科学研究センターにある GPU クラスタ HA-PACS のベースクラスタ部で行った．表 3.1 に HA-PACS ベースクラスタ部の性能諸元を，表 3.2 にソフトウェア環境を示す．

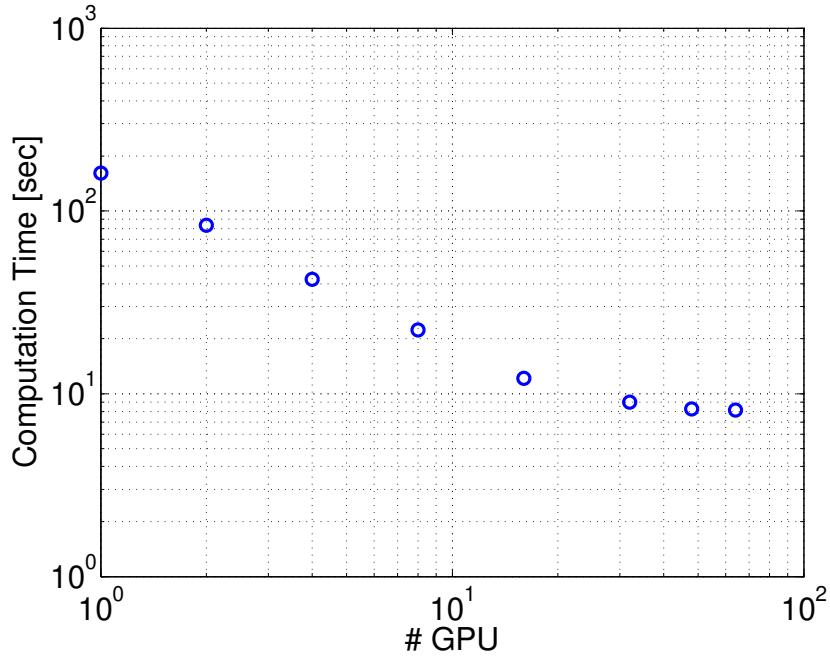

図 3.1　数値実験 3-1: 計算時間

3.4.1　数値実験 3-1

数値実験 3-1 では，使用ノード数及び GPU 数に対する提案実装のスケーラビリティを評価する．実験は，予め MATLAB で乱数によって生成した 10000 次元の実対称定値一般化固有値問題に対して，実対称定値一般化固有値問題に特化した実装を適用し，計算時間の計測を行った．Sakurai-Sugiura 法のパラメータは，$N = 32, M = 8, L = 32, \alpha = 0.1, \delta = 10^{-14}$ とし，積分路は固有対が全体の 1% が入るように設定した．使用するノード数は 1 から 16 まで変化させ，ノード内で使用する GPU の数は，16 ノード使用時のみ 1 から 4 の間で変化させた．

実験の結果を図 3.1 及び図 3.2 に示す．16GPU 以下ではノードあたり 1GPU を使用し，ノード数を変化させた場合を，16GPU 以上では 16 ノードでノードあたりの GPU を変化させた場合を表している．図 3.2 は，積分点の並列性が効く 16GPU まではほぼ理想的にスケールしている一方，線形ソルバの並列性を利用した 16GPU 以降はスケーラビリティが伸び悩んでいることを示している．これは，問題のサイズが小さいため，線形ソルバの並列性能が頭打ちになっているためであると考えられる．

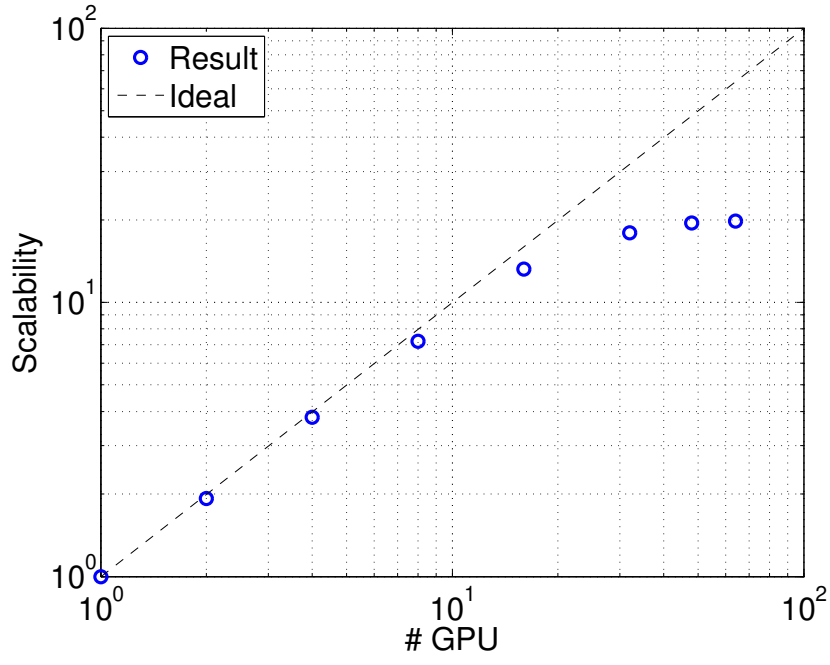

図 3.2　数値実験 3-1: スケーラビリティ

3.4.2　数値実験 3-2

数値実験 3-2 では，問題の大きさによって提案実装のパフォーマンスがどのように変化するのかを評価する．問題は数値実験 3-1 と同様の問題を，5,000 次元，15,000 次元，20,000 次元において作成し，それぞれ実対称定値一般化固有値問題向け実装を適用し，計算時間の計測を行った．各大きさの問題について，固有対がそれぞれ全体の 1% 入るように周回路を設置した．Sakurai-Sugiura 法のパラメータ L は，固有対の相対残差ノルムが十分小さくなるように設定した．16 ノード使用し，ノード内の GPU は 4GPU 使用した．

図 3.3，及び図 3.4 に結果を示す．なお，LU は LU 分解にかかる時間，solve は前進後退代入にかかる時間，calc S は $\hat{S}$ の計算にかかる時間，SVD は $\hat{S}$ の特異値分解の計算にかかる時間，RR は射影以降固有対を抽出するための計算時間，misc は関数呼び出しのコストなど雑多な計算時間をそれぞれ示している．図 3.4 は，問題の大きさを変化させても一つの周回路において計算する固有対の割合が変化しない場合，各々の部分における計算時間の割合が変化していないことを示している．

3.4.3　数値実験 3-3

数値実験 3-3 では，一つの周回路で計算する固有値の数が提案実装のパフォーマンスにどのような影響を与えるのか評価する．問題は数値実験 3-1 と同一のものを使用し，ノード数は 16，ノード内で

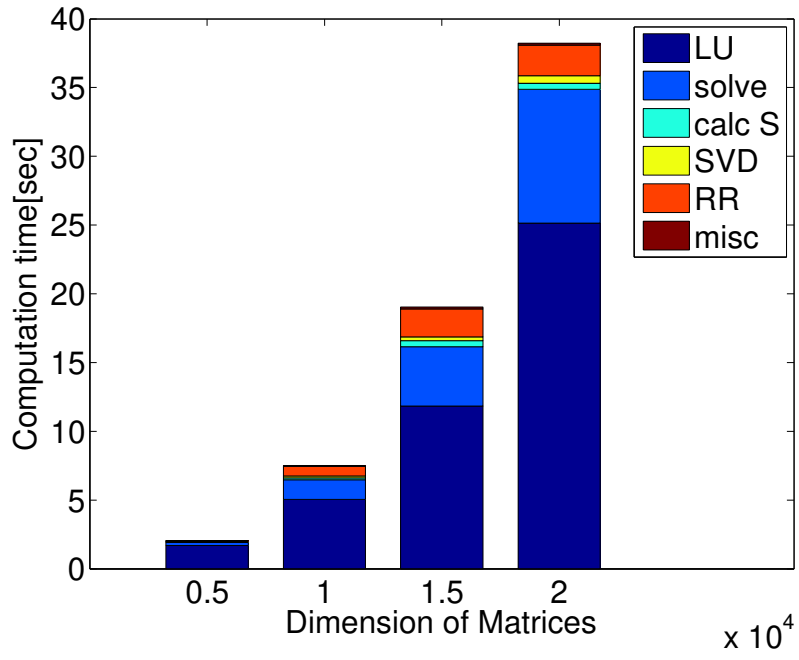

図 3.3　数値実験 3-2: 計算時間

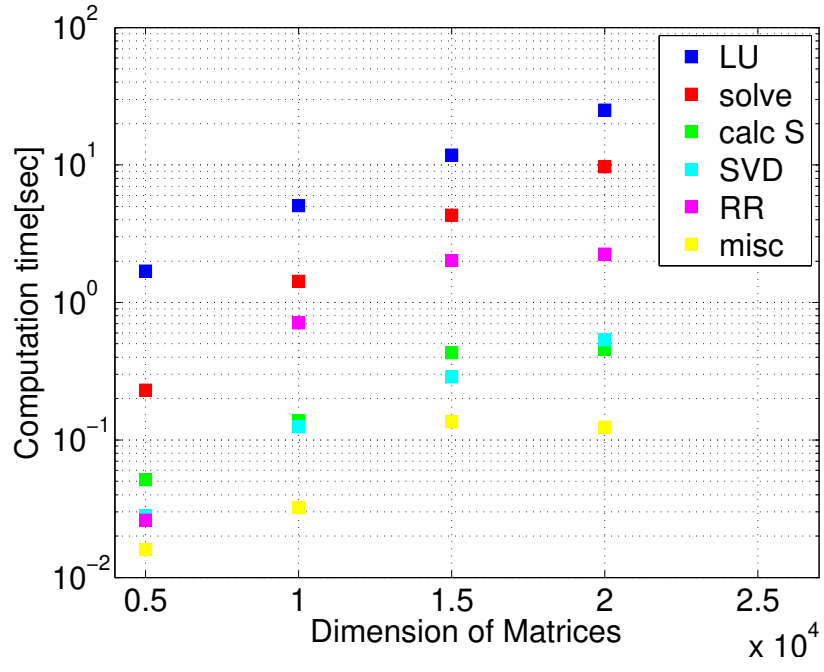

図 3.4　数値実験 3-2: 各パートごとの計算時間

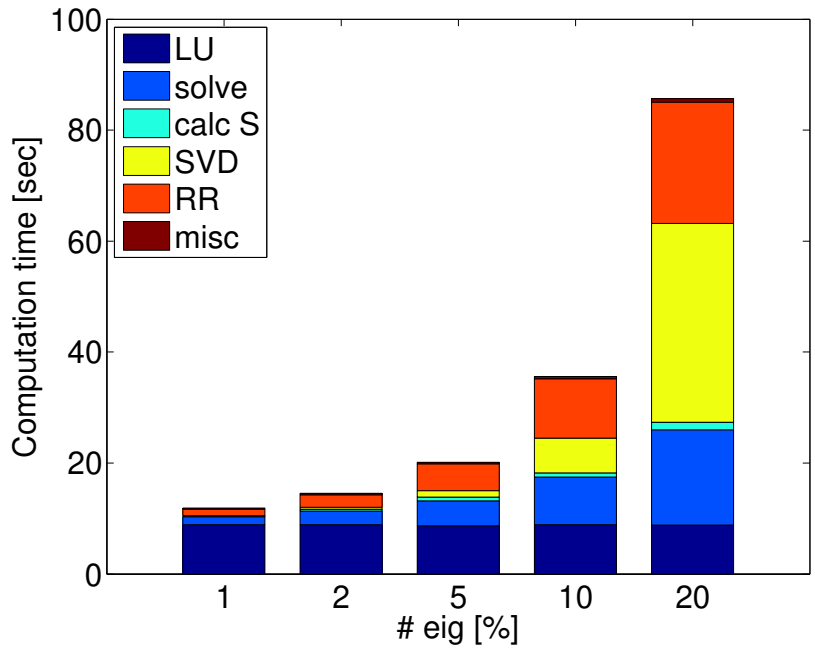

図 3.5　数値実験 3-3: 計算時間

使用する GPU の数は 4 とした．周回路に含まれる固有対の数を，$1\%, 2\%, 5\%, 10\%, 20\%$ となるように周回路を設定し，Sakurai-Sugiura 法のパラメータ L は残差ノルムが十分小さくなるようにそれぞれ $32, 64, 128, 256, 512$ と設定した．

図 3.5 及び図 3.6 に結果を示す．結果から，L を大きく取り過ぎると $\hat{S}$ の特異値分解や射影計算の時間が増大する傾向が伺える．従って，多数の固有対を計算する場合，一つの周回路で計算するよりも，領域を分割し複数の領域で計算を行ったほうが良いと考えられる．

3.4.4　数値実験 3-4

数値実験 3-4 では，応用分野で実際に現れる問題に対して提案実装を適用し，実用十分な精度で固有対を求めることが可能であるか評価する．問題は，ELSES matrix library[13, 17] より，AUNW9180[16] を選択した．この問題は電子状態計算で現れる 9180 次元の実対称行列の一般化固有値問題であり，バンドギャップ付近のおよそ 1% の固有対を求める必要がある．この問題に対して，実対称定値一般化固有値問題向け実装を適用した．16 ノード使用し，ノード内で使用する GPU の数は 4 とした．Sakurai-Sugiura 法のパラメータ L は 25 とした．

計算は，6.670[sec] で終了し，固有値及び相対残差ノルム

$$\frac{\|A\boldsymbol{x}_l - \lambda_l B\boldsymbol{x}_l\|}{\|A\boldsymbol{x}_l\| + \lambda_l \|B\boldsymbol{x}_l\|}$$

は図 3.7 の通りとなった．このことから，提案実装は実際に応用分野で現れる問題を精度良く解くこ

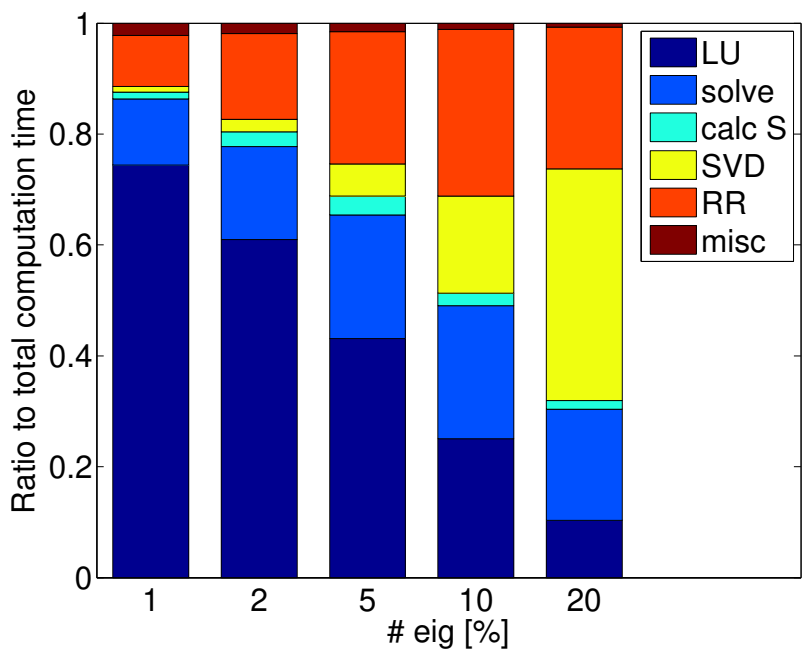

図 3.6　数値実験 3-3: 各パートが全体に占める割合

とができたと考えられる.

3.5　小括

本章では，GPU クラスタにおける中規模密行列の一般化固有値問題ソルバとして，MPI と線形計算ライブラリ MAGMA および cuBLAS を使用した Sakurai-Sugiura 法の分散並列実装を提案した. そして，数値実験により 2 万次元程度の密行列に対して高性能・高スケーラビリティを実現することができる実装であることを確認した.

MAGMA の直接法線形ソルバはノード内の GPU しか利用できないことから，より大きな問題に対して適用するためには分散並列実行可能な線形ソルバが必要である.

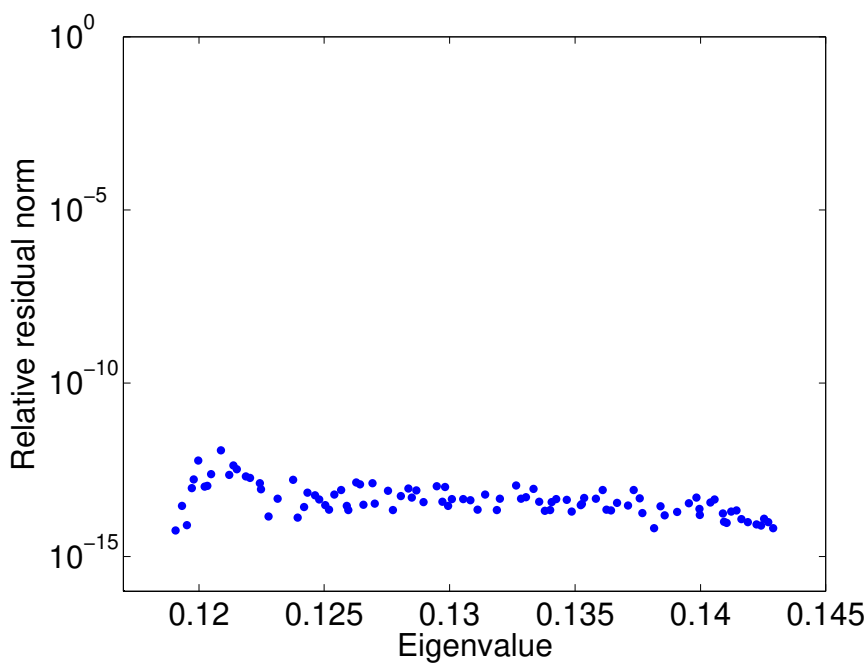

図 3.7　数値実験 3-4: 求めた固有値と相対残差ノルム

第 4 章

大規模密行列向け block Krylov 部分空間反復法の分散並列実装

前節の実装手法は，MAGMA を用いた連立一次方程式の求解において技術的制約により大規模な行列に対して適用することができないという問題点があった．block Krylov 部分空間反復法は複数右辺ベクトルを持つ連立一次方程式を効率よく解くことができる反復法の一種であり，反復過程において行列-複数右辺ベクトル積の計算が必要である．メニーコア環境における計算ノードでは従来の環境における計算ノードに比べ高速に密行列の行列-行列積を計算することが可能である．そのため，主に大規模疎行列向けに研究が行われている block Krylov 部分空間反復法を密行列向けに実装することで，計算ノードの性能を十分に引き出す事が可能な連立一次方程式解法の分散並列実装を構築することができると考えられる．本節では block Krylov 部分空間反復法の密行列向け分散並列実装を提案する．

4.1　密行列に対して反復線形ソルバを導入する動機

密行列に対して block Krylov 部分空間反復法を適用する場合，反復過程における主要な計算カーネルは $n \times n$ 行列と $n \times L$ 行列における，分散行列-行列積 (GEMM) となる．GEMM はメニーコア環境において高い性能を発揮することができ，LU 分解や前進後退代入に比べて高いスケーラビリティを発揮すると考えられる．このため，逐次実行においては計算量の面から block Krylov 部分空間反復法は直接法線形ソルバに比べて計算時間が遅くなると考えられるが，多並列時での計算時間はそのスケーラビリティの差により block Krylov 部分空間反復法の方が速くなると考えられる．また，反復法であることから連立一次方程式の解の精度を調整することが可能である．一つの積分点における連立一次方程式の解が誤差を持っている場合に Sakurai-Sugiura 法で固有対を計算すると，正確な連立一次方程式の解を用いて計算される固有対に比べて低い精度の固有対が計算されることが誤差解析の結果 [21] から知られている．加えて，すべての積分点において連立一次方程式の解に同程度の誤差が含まれている場合にも，正確な連立一次方程式を使用して計算された固有対よりも精度の低い固有値を計算されることが経験的に知られている．このことから，もし精度の低い固有対が計算される

Algorithm 5 Block BiCGrQ for solving $AX = B$ and $A^{\mathrm{H}}\tilde{X} = B$

Input: $A, X_0, B, \tilde{R}_0$

Output: $X_k, \tilde{X}_k$

1: $[Q_0, C_0] = \mathrm{qr}\left(B - AX_0\right), \quad \left[\tilde{Q}_0, \tilde{C}_0\right] = \mathrm{qr}\left(\tilde{R}_0\right)$
2: $V_0 = Q_0, \tilde{V}_0 = \tilde{Q}_0$
3: **for** $k = 0, 1, \ldots,$ **do**
4: $\quad T_k = \left(\tilde{V}_k^{\mathrm{H}} A V_k\right)^{-1} \left(\tilde{Q}_k^{\mathrm{H}} Q_k\right), \quad \tilde{T}_k = \left(V_k^{\mathrm{H}} A^{\mathrm{H}} \tilde{V}_k\right)^{-1} \left(Q_k^{\mathrm{H}} \tilde{Q}_k\right)$
5: $\quad X_{k+1} = X_k + V_k T_k C_k, \ \tilde{X}_{k+1} = \tilde{X}_k + \tilde{V}_k \tilde{T}_k \tilde{C}_k$
6: $\quad [Q_{k+1}, S_{k+1}] = \mathrm{qr}\left(Q_k - A V_k T_k\right), \quad \left[\tilde{Q}_{k+1}, \tilde{S}_{k+1}\right] = \mathrm{qr}\left(\tilde{Q}_k - A^{\mathrm{H}} \tilde{V}_k \tilde{T}_k\right)$
7: $\quad W_k = \left(\tilde{Q}_k^{\mathrm{H}} Q_k\right)^{-1} \tilde{S}_{k+1}^{\mathrm{H}} \left(\tilde{Q}_{k+1}^{\mathrm{H}} Q_{k+1}\right), \quad \tilde{W}_k = \left(Q_k^{\mathrm{H}} \tilde{Q}_k\right)^{-1} S_{k+1}^{\mathrm{H}} \left(Q_{k+1}^{\mathrm{H}} \tilde{Q}_{k+1}\right)$
8: $\quad V_{k+1} = Q_{k+1} + V_k W_k, \ \tilde{V}_{k+1} = \tilde{Q}_{k+1} + \tilde{V}_k \tilde{W}_k$
9: $\quad C_{k+1} = S_{k+1} C_k, \ \tilde{C}_{k+1} = \tilde{S}_{k+1} \tilde{C}_k$
10: **end for**

ことが許容される場合，固有対の精度と引き換えに連立一次方程式の収束判定条件をより緩く設定することで逐次計算量を削減することが可能である．

一方で，反復解法であることから問題によっては解が収束判定条件を満たせずに収束しない場合もあることや，設定する周回路，積分点毎に反復回数が違うことが考えられるため，Top Layer および Middle Layer に関する並列性においてロードバランスが崩れるという問題点が挙げられる．

Krylov 部分空間反復法は前処理によってその収束性能が大きく変化する．しかし，前処理を適用することは本稿では扱わず，今後の課題とする．

4.2 BiCG 型解法の採用

2.2.4 節における関係 (2.15) を利用すると，適切な初期解を設定した上で双 Lanczos 基底を用いた block Krylov 部分空間反復法を採用すると，C_j および C_{N-j+1} に関する連立一次方程式を同時に解くことが可能である．この性質により，block BiCG 型の解法を採用することで Sakurai-Sugiura 法で現れる連立一次方程式を効率よく解くことが可能である．本稿では，反復毎に残差行列を直交化することにより数値的安定性を高めた block BiCGrQ 法 [10, Algorithm 6] を使用することを提案する．**Algorithm 5** に block BiCGrQ 法のアルゴリズムを示す．

4.3 分散並列実装

反復計算の主要部である $n \times n$ 行列に関する分散 GEMM のデータ分散形式として，ScaLAPACK で用いられている 2D block cyclic distribution ではなく，2D block distribution を採用する．2D block distribution を採用する理由として，通信パターンの煩雑化の回避および通信回数の削減が挙

Algorithm 6 2D block distribution における分散行列-行列 (複数本ベクトル) 積の計算手順

1: Perform local matrix-matrix multiplication $A_{ij}V_j$	`*GEMM`
2: (All)reduce the local product on P_{i*}	`MPI_Reduce`
3: Broadcast the result from P_{jj} to P_{*j}	`MPI_Bcast`

げられる．また，行列の分解を行わないため，計算負荷の著しい偏りが生じないことも挙げられる．本節では，block BiCGrQ 法を構成するために必要な各種計算の分散並列実装について述べる．

4.3.1 2D block distribution による分散行列-行列積

行列 $A \in \mathbb{C}^{n\times n}$ および L 本のベクトルからなる行列 $V = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_L] \in \mathbb{C}^{n\times L}$ の積 AV を p^2 プロセスを用いて計算することを考える．p^2 プロセスからなるプロセスグリッドを考え，各プロセスを $P_{ij}, 1 \leq i, j \leq p$ と表現する．プロセスグリッド上での行列 A の 2D block distribution を

$$A = \begin{bmatrix} A_{11} & \cdots & A_{1p} \\ \vdots & \ddots & \vdots \\ A_{p1} & \cdots & A_{pp} \end{bmatrix} \tag{4.1}$$

と表す．プロセス P_{ij} は A_{ij} を保持する．V の分散形式として以下の形式を考える．

$$V = \begin{bmatrix} V_1 \\ \vdots \\ V_p \end{bmatrix}. \tag{4.2}$$

ここで，V_j の行数は A_{ij} の列数と同じである．P_{ij} は V_j を保持する．左から A をかけると，

$$\begin{aligned} AV &= \begin{bmatrix} A_{11} & \cdots & A_{1p} \\ \vdots & \ddots & \vdots \\ A_{p1} & \cdots & A_{pp} \end{bmatrix} \begin{bmatrix} V_1 \\ \vdots \\ V_p \end{bmatrix} \\ &= \begin{bmatrix} A_{11}V_1 + \cdots + A_{1p}V_p \\ \vdots \\ A_{p1}V_1 + \cdots + A_{pp}V_p \end{bmatrix}, \end{aligned}$$

となる．つまり，プロセス P_{ij} は A_{ij} と V_j の積をローカルに計算を行い，P_{i*} と加算の (All)reduce を行うことで行列-行列積を計算することが可能である．この行列-行列積を一度行うと縦長行列の分散方向が反転するため，P_{jj} は P_{*j} へ計算結果を配布することで本来の分散方向に戻すことができる．

以上をまとめると，**Algorithm 6** となる．

Algorithm 7 一般化内積の計算手順

1: Perform local matrix-matrix multiplication $U_j^{\mathrm{H}} V_j$	`*GEMM`
2: Allreduce the local product on P_{i*}	`MPI_Allreduce`

4.3.2 一般化内積

L 本のベクトルからなる行列 $U, V \in \mathbb{C}^{n \times L}$ について，一般化内積 $U^{\mathrm{H}} V$ の計算を，分散行列-行列積で用いるデータ分散方法において行うことを考える．

$$
\begin{aligned}
W &:= U^{\mathrm{H}} V \\
&= \left[U_1^{\mathrm{H}}, \ldots, U_p^{\mathrm{H}}\right]^{\mathrm{H}} \begin{bmatrix} V_1 \\ \vdots \\ V_p \end{bmatrix} \\
&= U_1^{\mathrm{H}} V_1 + \cdots + U_p^{\mathrm{H}} V_p,
\end{aligned}
$$

であるから，プロセス P_{*j} は $U_j^{\mathrm{H}} V_j$ の小行列積をローカルで行った後，プロセスグリッドの同じ行に属するプロセス間で計算結果に対して加算の Allreduce を行うことで一般化内積を計算することができる．

計算手順をまとめると，**Algorithm 7** となる．

4.3.3 直交化

行列の thin QR 分解を計算する手法として，Gram-Schmidt 法や数値的安定性が高い Householder QR 法などが提案されている．本研究では，実装の簡便性の観点から残差行列の直交化手法として Cholesky QR 法 [32] を採用した．

Cholesky QR 法は，分解の対象となる縦長行列 $V \in \mathbb{C}^{n \times L}$ の Gram 積 $W := V^{\mathrm{H}} V$ の Cholesky 分解を利用して QR 分解を計算する．このため，V の列ベクトルが線形従属に近い場合は Cholesky 分解が計算できないことにより計算が破綻する可能性があることに注意する．

実装は前小節の一般化内積を用いて Gram 積を計算した後，各プロセスにおいて Cholesky 分解をローカルに計算し，各プロセスが保持する縦長行列の小行列との積をローカルに行う．

Cholesky QR 法のアルゴリズムおよび使用する関数を **Algorithm 8** に示す．

4.4 数値実験

本節では提案した block BiCGrQ 法の分散並列実装および Sakurai-Sugiura 法へ組み込んだ場合の性能を，応用で実際に現れる問題を用いた数値実験により評価する．数値実験は JCAHPC が運営する大規模 Xeon Phi クラスタである Oakforest-PACS を最大で 1024 ノード使用して行った．各計算ノードには Knights Landing 世代の Xeon Phi が 1 基搭載されており，メモリは DDR4 が

Algorithm 8 Cholesky QR 法のアルゴリズムと使用する関数

Input: $V \in \mathbb{C}^{n \times L}$

Output: $Q \in \mathbb{C}^{n \times L}, R \in \mathbb{C}^{L \times L}$

1: Compute Gram product $W := V^{\mathrm{H}}V$	**Algorithm 7**
2: Perform Cholesky factorization $W = R^{\mathrm{H}}R$	`ZPOTRF`
3: Perform local matrix-matrix multiplication $Q_j = V_j R^{-1}$	`ZTRTRI,ZGEMM`

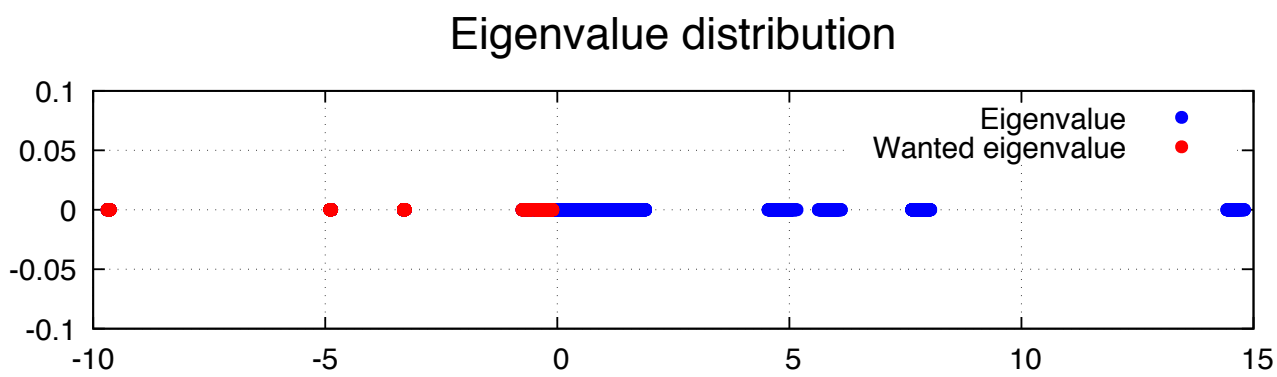

図 4.1　対象とする問題の固有値分布

96GB と MCDRAM が 16GB 利用可能である．本稿の実験では MCDRAM は cache として利用した．また，1 ノードあたり 4 プロセスを割り当て，各プロセスに 64 個の OpenMP スレッドを KMP_AFFINITY=compact で割り当てた．また，計算では動的 Tickless 設定コアのみを使用した．

実験で用いた問題は電子状態計算プログラム SIRIUS[31, 22] から得られた $n = 95951$ の行列からなるエルミート定値一般化固有値問題である．図 4.1 に固有値の分布を示す．この行列の区間 $[-10, -0.1]$ 内にある 1960 個の固有値および固有ベクトルを計算することを考える．

Sakurai-Sugiura 法でこの区間の固有対を 1 区間として計算することは困難であることから，$[-10, -9], [-5, -4], [-4, -3], [-1, -0.1]$ の 4 区間を設定した．以降，これらを区間 1,2,3 および 4 と呼ぶこととする．図 4.2 に，各区間における積分点の配置を示す．数値実験 4-1 を除いて，各区間において Sakurai-Sugiura 法のパラメータは $N = 16, M = 8, L = 256$ とした．

block BiCGrQ 法を Sakurai-Sugiura 法へ組み込む際に，Sakurai-Sugiura 法のライブラリである z-Pares を利用した．z-Pares 自体は Middle Layer および Bottom Layer の並列性を利用可能であるが，実験では Bottom Layer の並列性のみを使用した．

4.4.1　数値実験 4-1

数値実験 4-1 では，block BiCGrQ 法の各積分点における反復回数および提案した実装手法による総計算時間がパラメータ L によってどのように影響を受けるか検証する．本実験では Sakurai-Sugiura 法に対して適用した際に，計算が破綻すると考えられる比較的小さな L に対しても，線形ソルバとしての検証のために調査を行う．

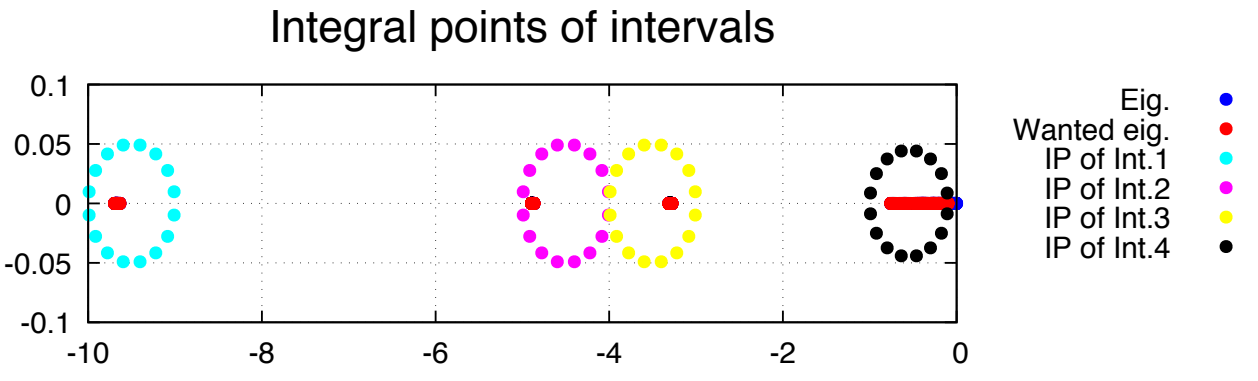

図 4.2　設置した区間と積分点

パラメータ L を $2^4, 2^5, \ldots, 2^{10}$ と設定し，各条件において各区間・積分点における block BiCGrQ 法の反復回数および全積分点の総計算時間を計測した．使用ノード数は 64 ノードで計測を行った．Block BiCGrQ 法の最大反復回数を 1000 回，収束判定条件は 10^{-6} に設定した．

各区間における積分点毎の反復回数を図 4.3 から図 4.6 に，区間毎の総計算時間を図 4.7 に，区間 4 での計算時間を元に算出した block BiCGrQ の反復における各計算部分の計算時間およびその割合を図 4.8 および図 4.9 に，そして計算の主要部となる分散 GEMM の各処理にかかる時間およびその割合を図 4.10 および図 4.11 に示す．

図 4.3，図 4.4，図 4.3 および図 4.4 から L を増加させると，block BiCGrQ 法の反復回数は減少していることが確認できる．しかしながら図 4.7 では，L を増加させると総計算時間はおおよそ増加している．これは，L の増加による反復回数の減少の割合に比べて反復毎の計算時間の増加の割合の方が計算時間に対する寄与が大きいためであると考えられる．図 4.8 から，L の増加に対して分散 GEMM の計算時間がほぼ線形に増加していることが確認できる．一方で，図 4.9 から，L が増加すると block BiCGrQ 法の反復における分散 GEMM の占める割合は減少していることが確認できる．これは，一般化内積など L が線形以上で計算量に寄与する計算に関する計算時間の増大のためであると考えられる．図 4.10 において，$L = 256, 1024$ においてローカルの GEMM の計算の計算時間の傾向が変化しているように見られる．これは，内部で使用している MKL BLAS が GEMM のサイズによって内部で使用している kernel を変えているためであると考えられる．

4.4.2　数値実験 4-2

数値実験 4-2 では提案した block BiCGrQ 法と ScaLAPACK の直接法線形ソルバである PZGETRF および PZGETRS のスケーラビリティの比較を行った．数値実験 4-2 は数値実験 4-3 の予備実験であると同時に，両線形ソルバが Sakurai-Sugiura 法の Bottom Layer の並列性をどの程度活用できるのかを調査する．

Block BiCGrQ 法について，一つの積分点における連立一次方程式求解にかかる時間を，$4^2, 8^2, 12^2, \ldots, 32^2$ ノードを使用して計測した．全領域の各積分点において予め反復回数を調べておき，一番反復回数が多かった積分点を計測対象の積分点として使用した．使用するノード数が平方数

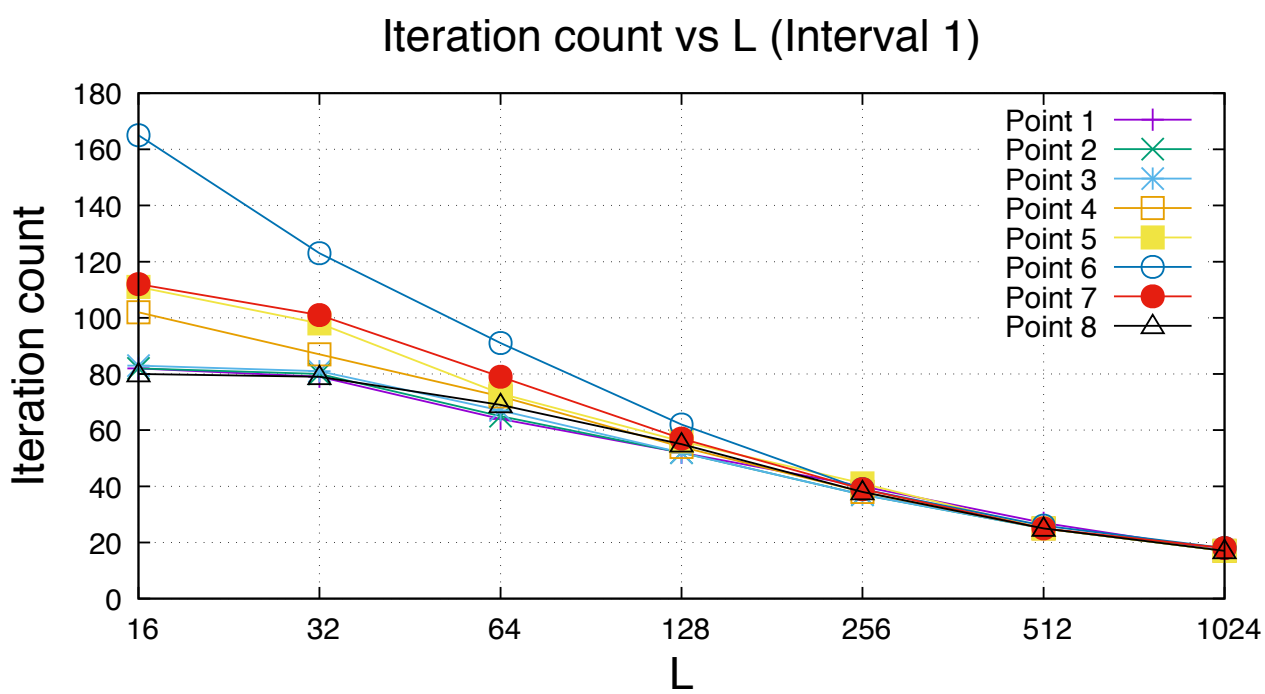

図 4.3　区間 1 の各積分点における L に対する block BiCGrQ 法の反復回数

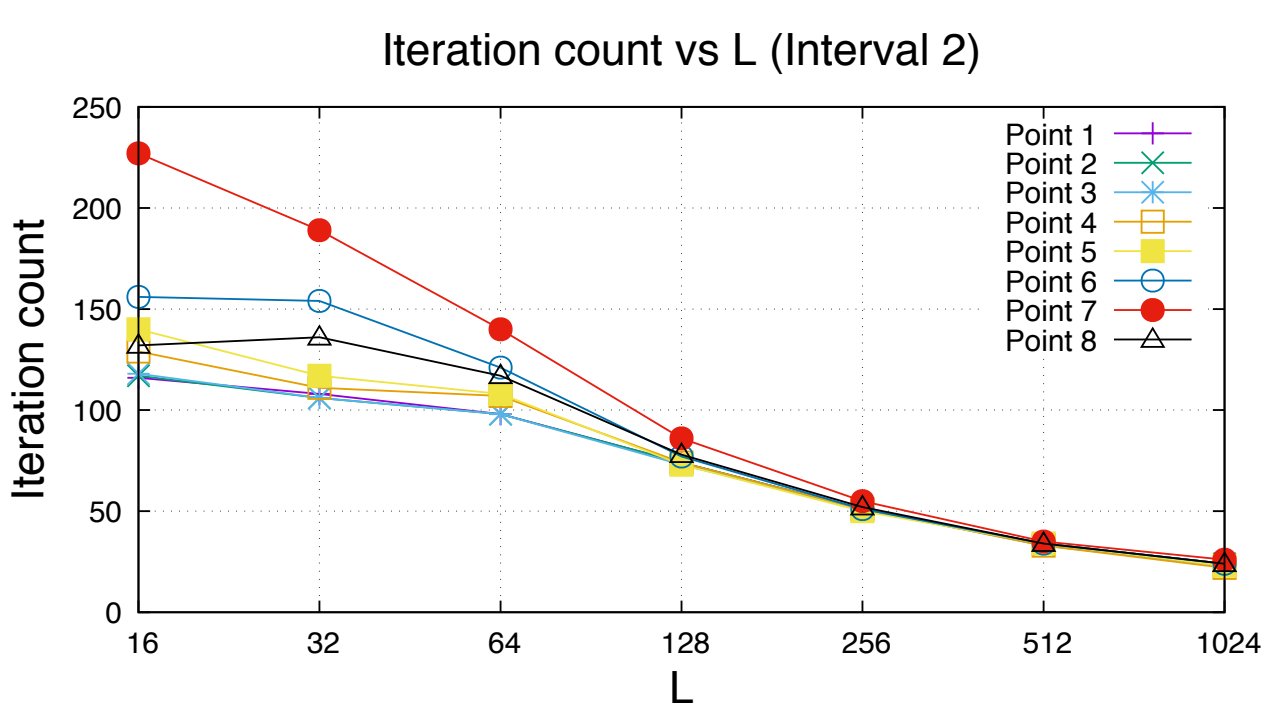

図 4.4　区間 2 の各積分点における L に対する block BiCGrQ 法の反復回数

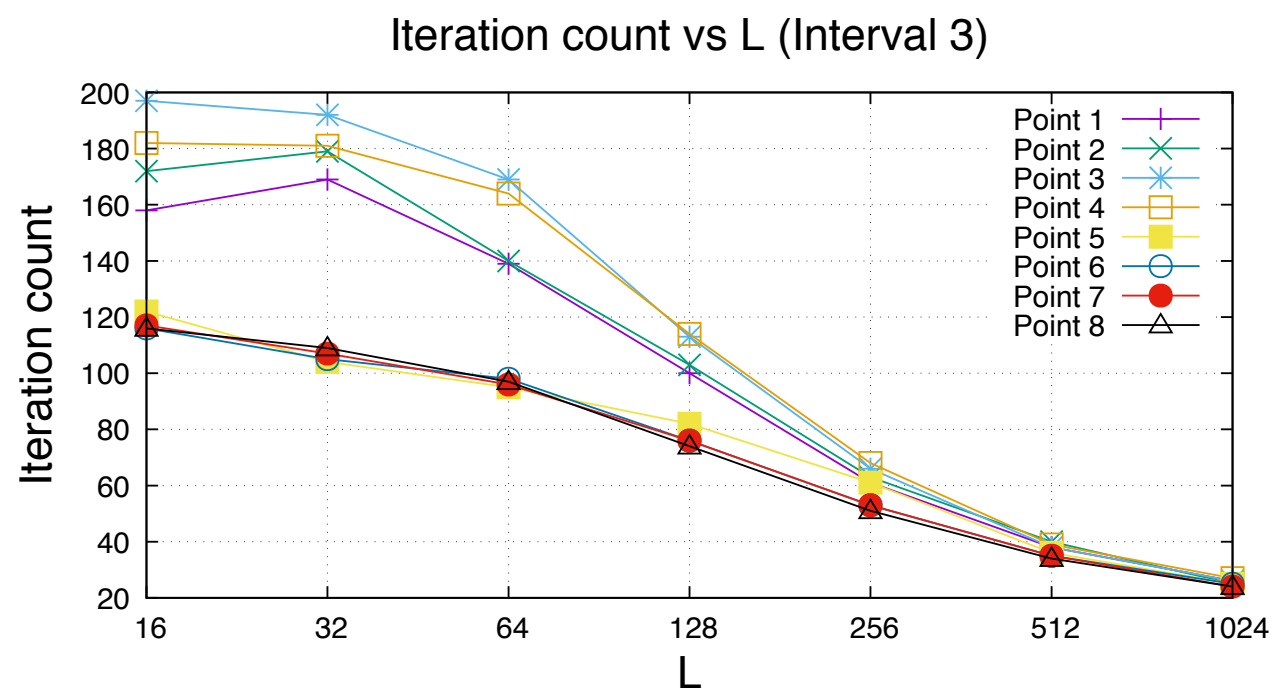

図 4.5　区間 3 の各積分点における L に対する block BiCGrQ 法の反復回数

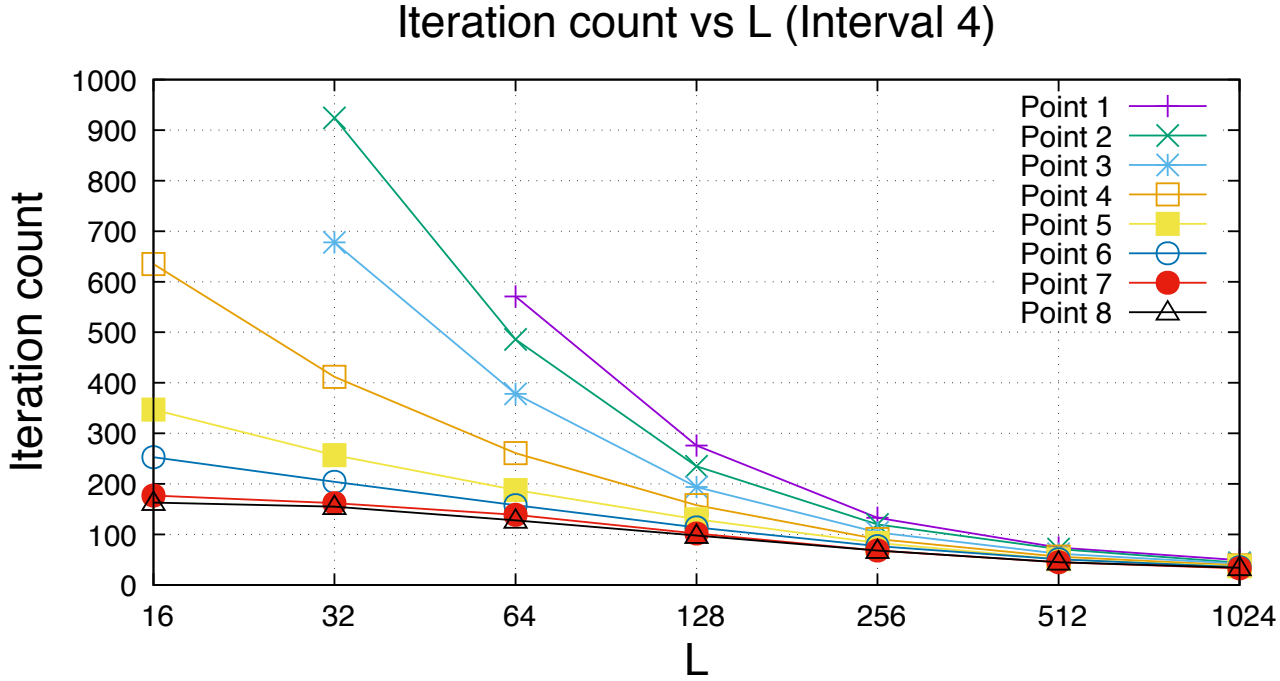

図 4.6　区間 4 の各積分点における L に対する block BiCGrQ 法の反復回数 ($L = 16$ における積分点 1,2,3 および $L = 32$ における積分点 1 は最大反復回数に到達)

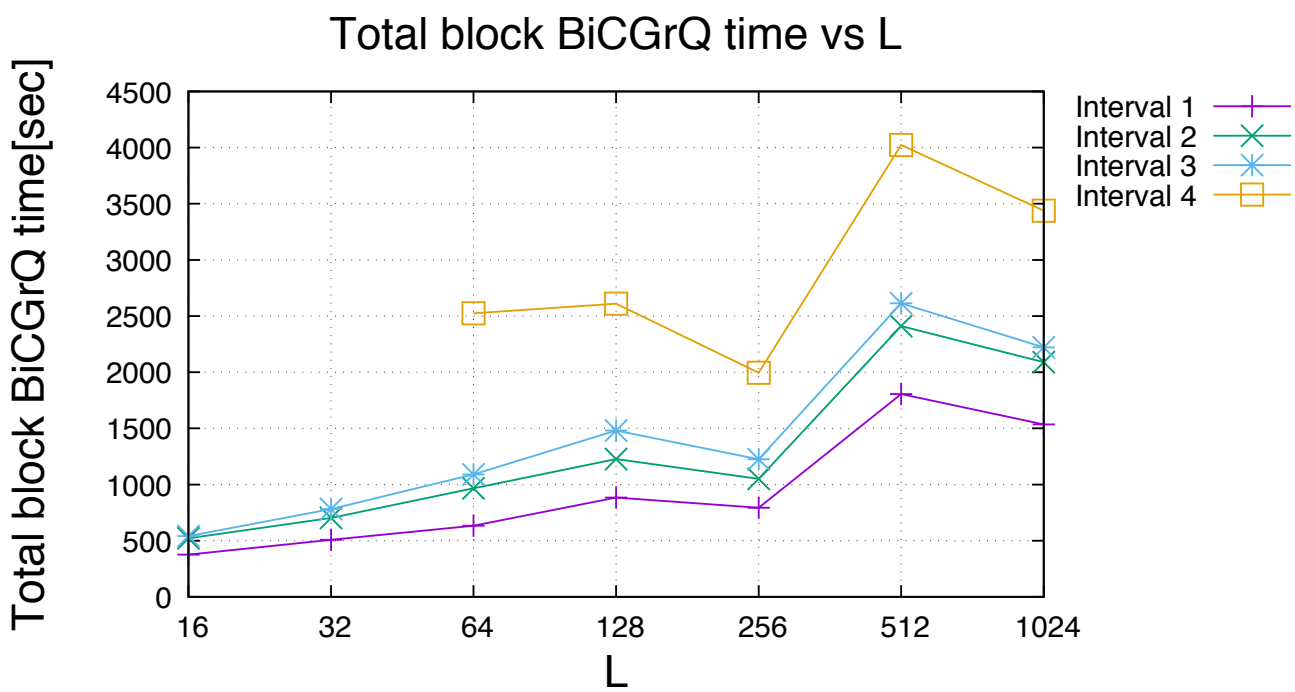

図 4.7　各領域の全積分点における block BiCGrQ 法の総計算時間 (区間 4 の $L = 16, 32$ は最大反復回数に到達した積分点が含まれるため除外)

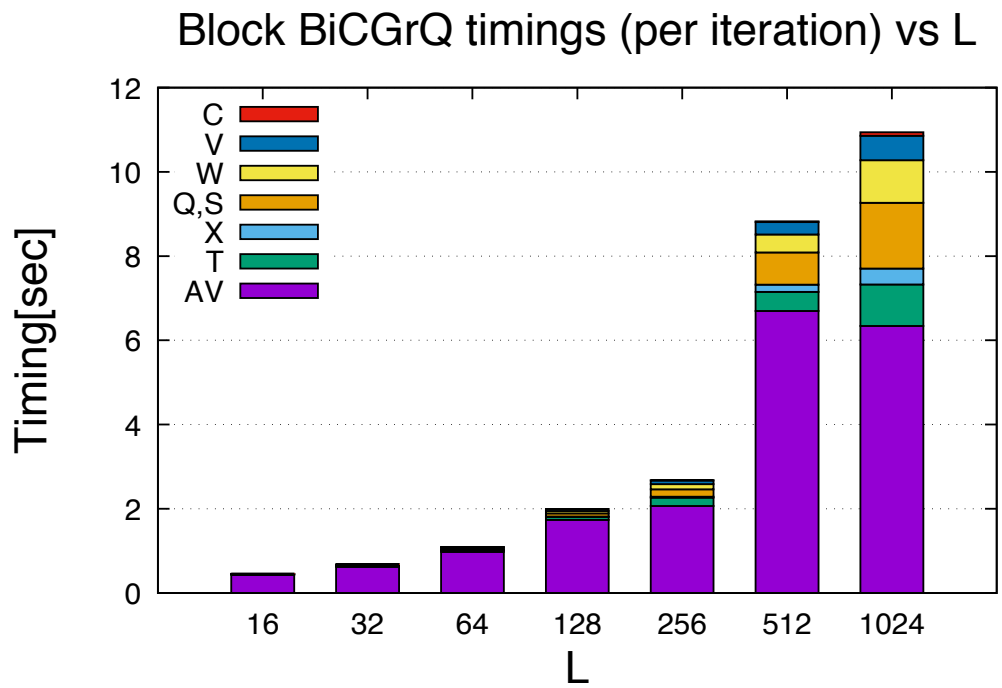

図 4.8　Block BiCGrQ 法の各計算部分のループ毎の計算時間

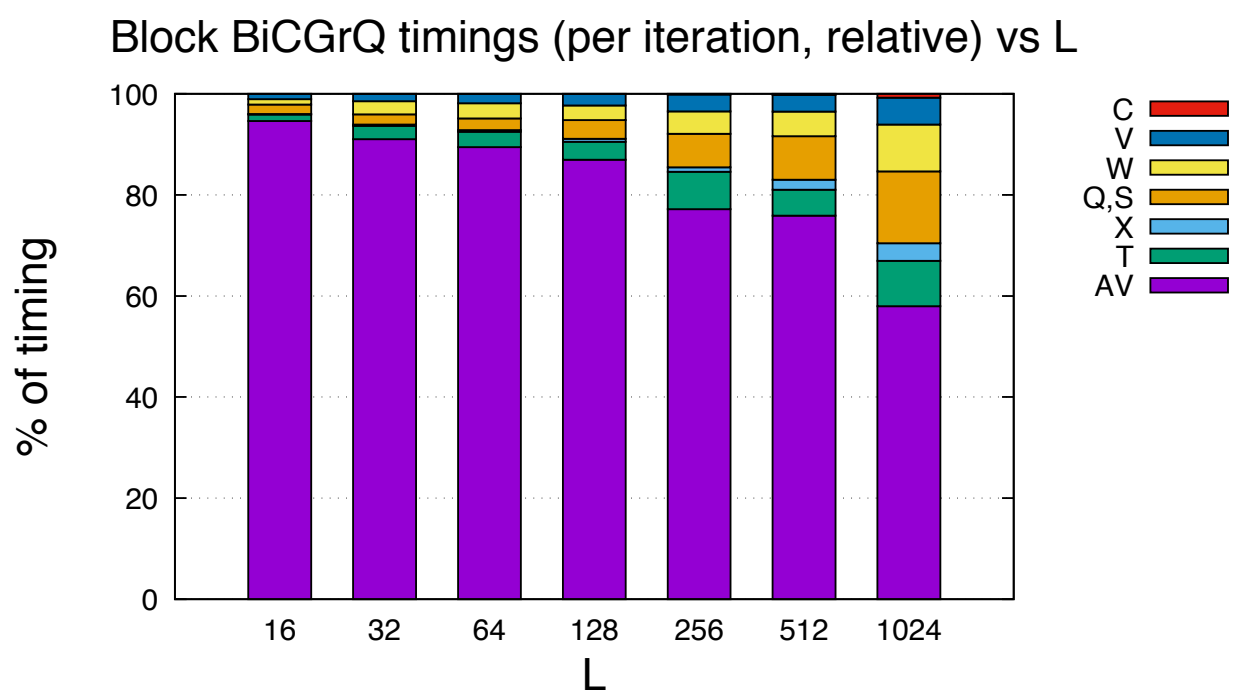

図 4.9　Block BiCGrQ 法の各計算部分のループ毎の計算時間の割合

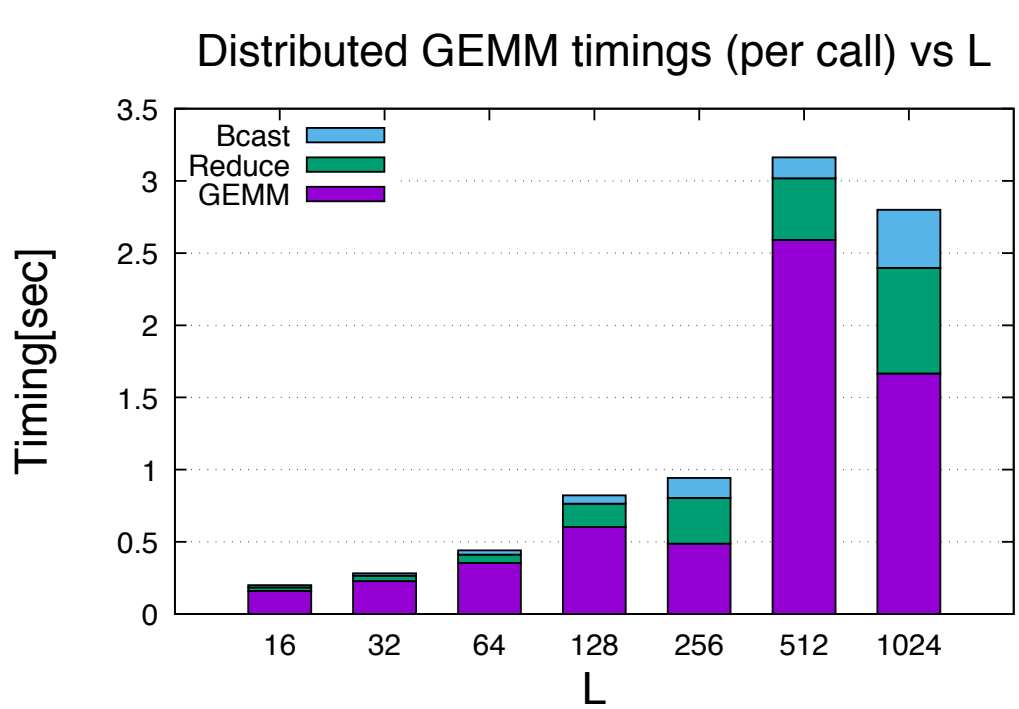

図 4.10　分散 GEMM の各計算部分の呼び出し毎の計算時間

となっているのは，分散 GEMM における行列の分割数を行方向と列方向で同一にするためである．収束判定条件は 10^{-6} とした．

ScaLAPACK の直接法について，1 つの積分点での連立一次方程式について 1 回の LU 分解および 2 回の前進後退代入を行う時間を計測した．計測に使用したノード数は block BiCGrQ 法と同一である．ScaLAPACK の 2D block cyclic distribution のブロックサイズ mb および nb は各々 1024 とした．

両線形ソルバについて 16 ノードでの計算時間を基準にスケーラビリティを算出した．図 4.12 に両者のスケーラビリティを，図 4.13 に ScaLAPACK の直接法線形ソルバの計算時間の内訳を，図 4.14 および図 4.15 に block BiCGrQ 法の計算時間の内訳およびその割合を，そして図 4.16 および図 4.17 に block BiCGrQ 法内部の分散 GEMM の計算時間の内訳およびその割合をそれぞれ示す．

図 4.12 から，block BiCGrQ は ScaLAPACK の直接法線形ソルバよりスケーラビリティが良好であることが分かる．図 4.13 を見ると，ScaLAPACK の LU 分解はあまりスケールせず，前進後退代入は全くスケールしない．これは，多数の OpenMP スレッド数が効いていることにより，ScaLAPACK の各関数におけるプロセス内の計算時間が通信が見える程度に小さくなっていることが原因である．一方，図 4.14 を見ると，block BiCGrQ 法は分散 GEMM のスケーラビリティが良く効いていることが確認できる．このため，Knights Landing 環境下では，密行列向け block BiCGrQ 法は ScaLAPACK の直接法線形ソルバより適した解法であると考えられる．しかしながら図 4.17 から，多ノード時においてはローカルの GEMM の計算時間よりも通信にかかる時間の方が多くなっていることから，これ以上線形ソルバとしての顕著な速度向上は見込めないと考えられる．

4.4.3 数値実験 4-3

数値実験 4-3 では，Sakurai-Sugiura 法に block BiCGrQ 法を組み合わせた固有値解法 (SSM-BB)，ScaLAPACK の直接法線形ソルバを組み合わせた固有値解法 (SSM-SL) および ScaLAPACK の直接法固有値解法の実装である PZHEGVX の 3 つの固有値解法の実装について，計算時間の比較を行う．SSM-BB および SSM-SL では数値実験 4-2 の結果を利用して総計算時間の推定を行った．

SSM-BB について，収束判定条件が $10^{-6}, 10^{-8}, 10^{-10}$ の場合について，64 ノードで全積分点の反復回数を予め計測し，計算時間推定に利用した．計算時間推定における各層の並列性の利用のされ方を表 4.1 に示す．ここで，$P_{\mathrm{int}}, P_{\mathrm{q}}, P_{\mathrm{LS}}, P_{\mathrm{total}}$ はそれぞれ上層，中層，下層の並列性において使用されるノードグループの数，および総利用ノード数である．Oakforest-PACS は 8208 ノードで構成されているが，本実験ではノード数が 32768 ノード以上あるものと仮定して計算時間の推定を行っている．

上層および中層の並列度を最大まで使い切った場合，P_{total} ノードにおける総計算時間は以下のように推定される．

$$T_{\mathrm{total}}(P_{\mathrm{total}}) := \max_{i,j} T_{\mathrm{LS}}^{(i,j)}(P_{\mathrm{LS}}) + T_{\mathrm{other}}(P_{\mathrm{LS}}).$$

ここで，$T_{\mathrm{LS}}^{(i,j)}(P_{\mathrm{LS}})$ および $T_{\mathrm{other}}(P_{\mathrm{LS}})$ は下層の並列数 P_{LS} における区間 i の j 番目の積分点における連立一次方程式求解時間および Sakurai-Sugiura 法の連立一次方程式求解以外にかかる時間であ

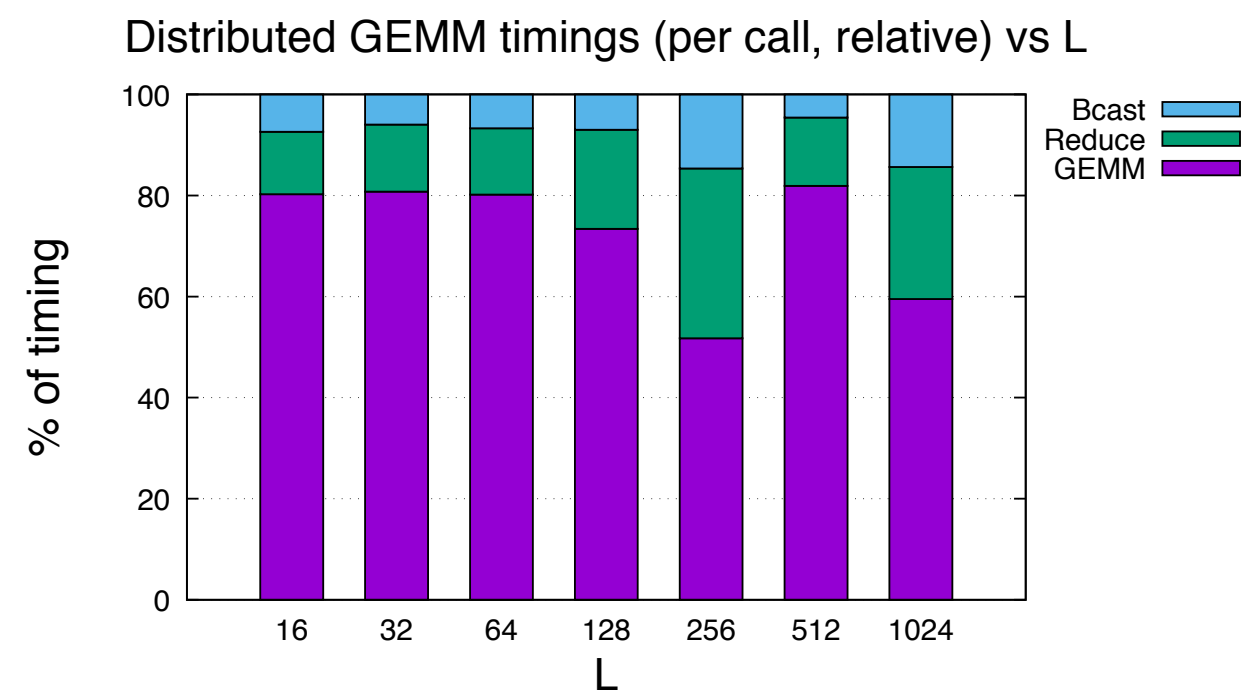

図 4.11　分散 GEMM の各計算部分の呼び出し毎の計算時間

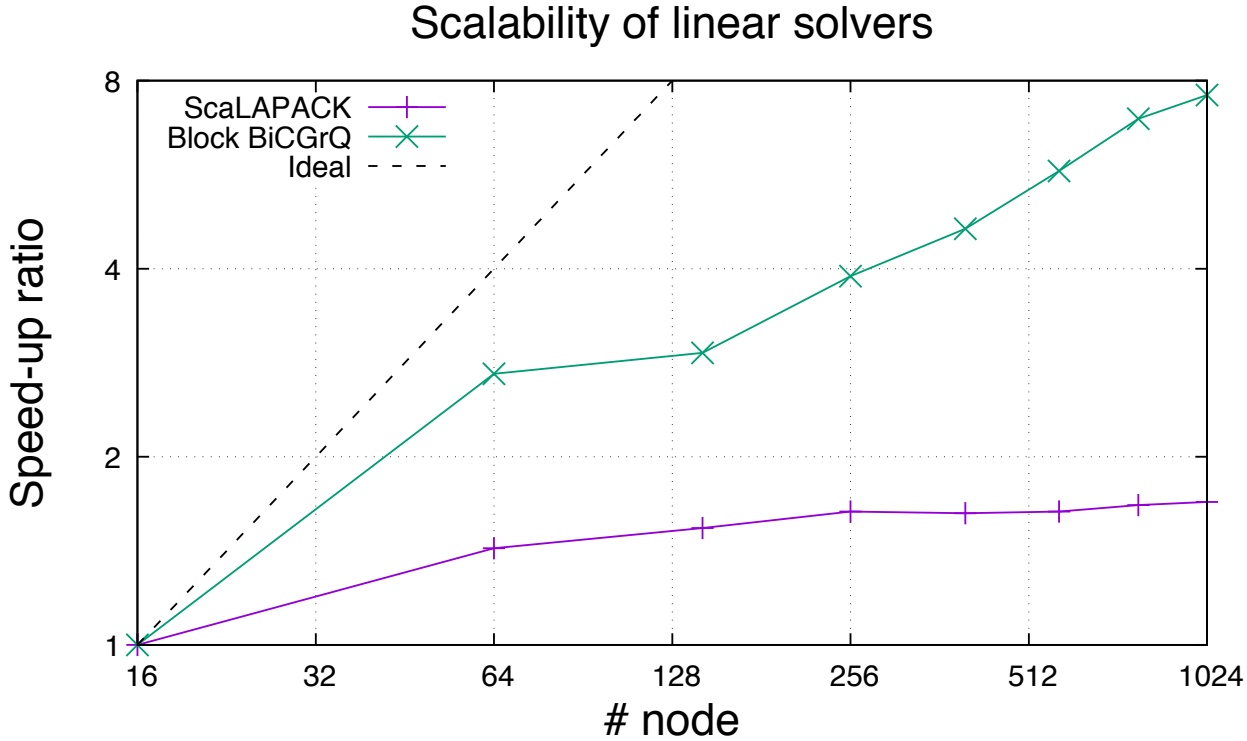

図 4.12　両線形ソルバのスケーラビリティ

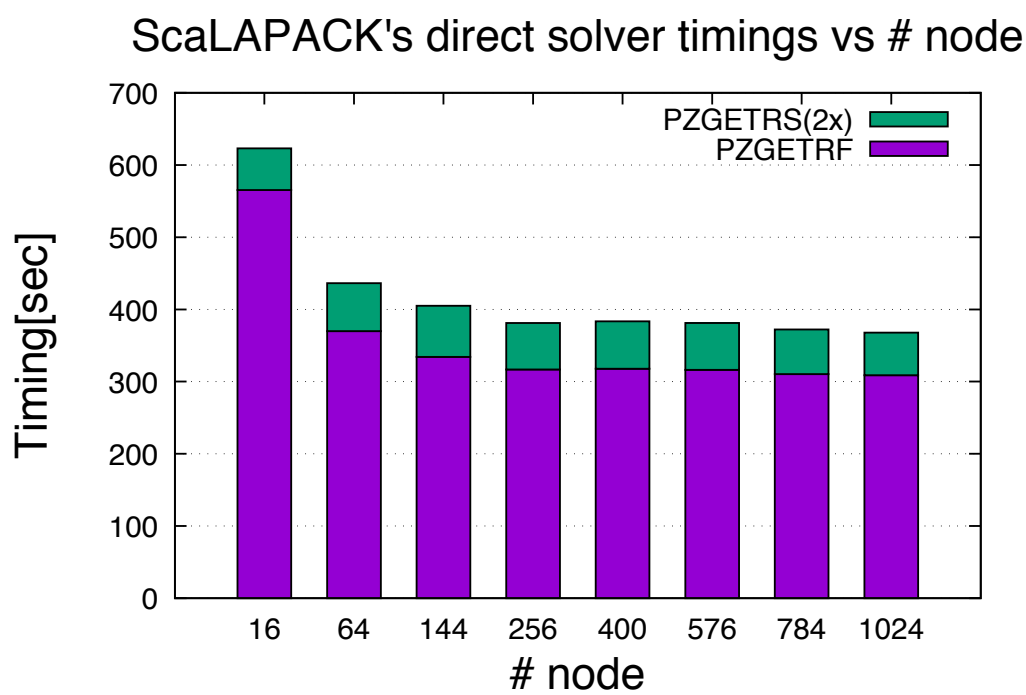

図 4.13　各ノード数における ScaLAPACK の直接法線形ソルバの計算時間内訳

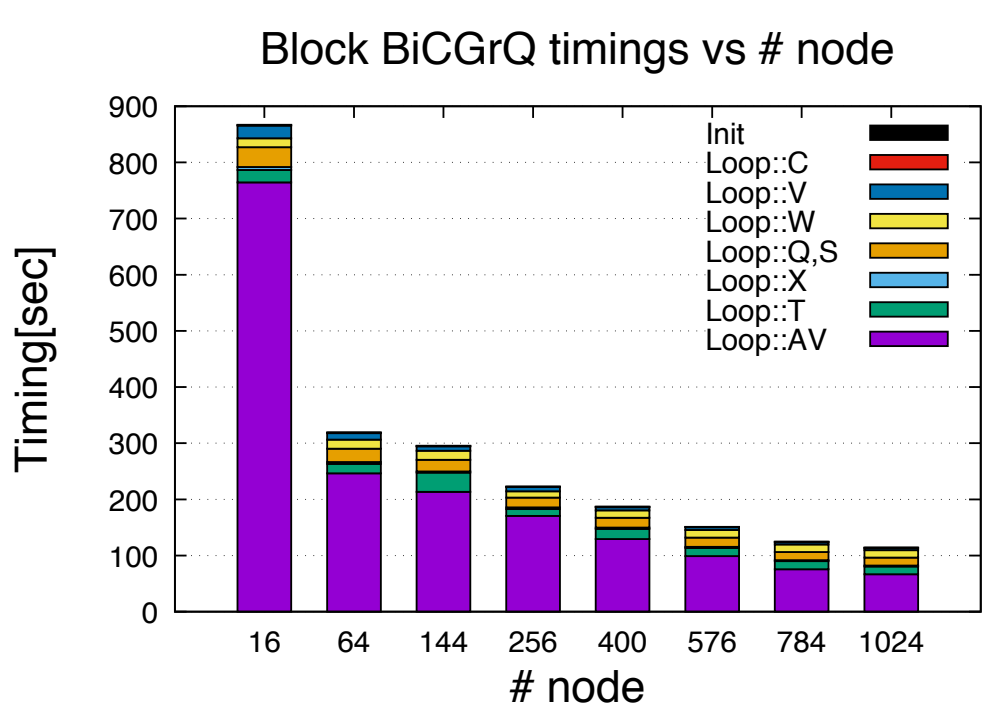

図 4.14　各ノード数における block BiCGrQ 法の計算時間内訳

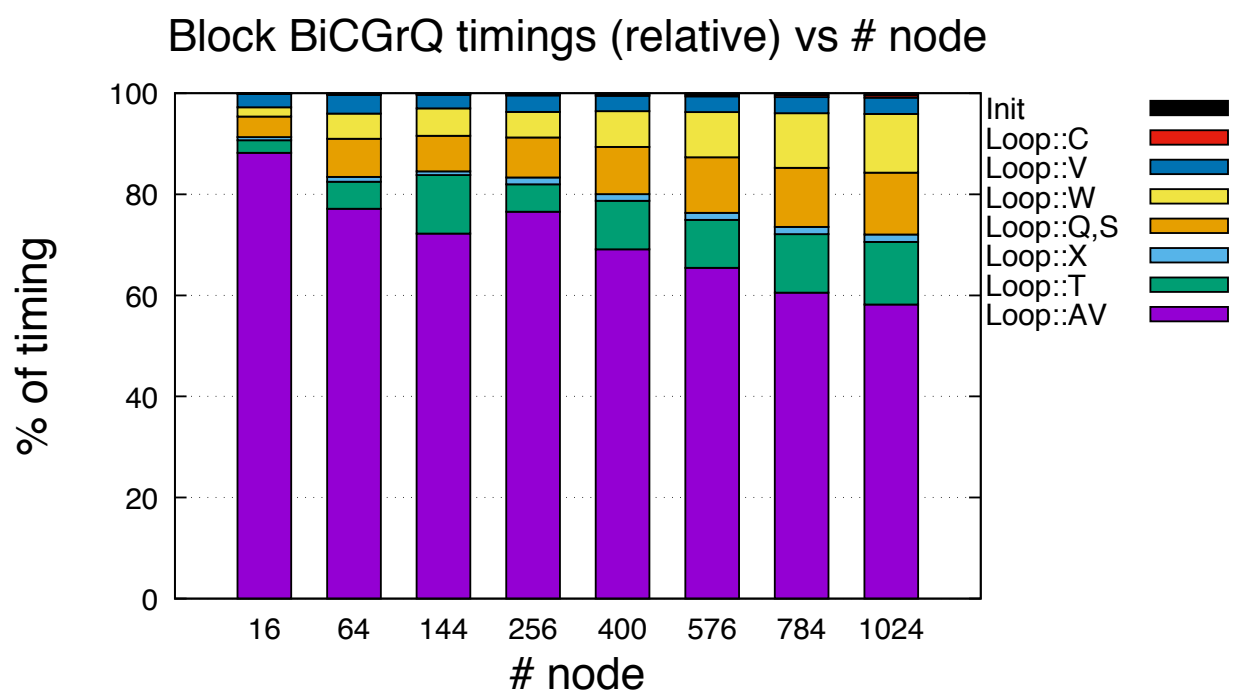

図 4.15　各ノード数における block BiCGrQ 法の各計算時間の割合

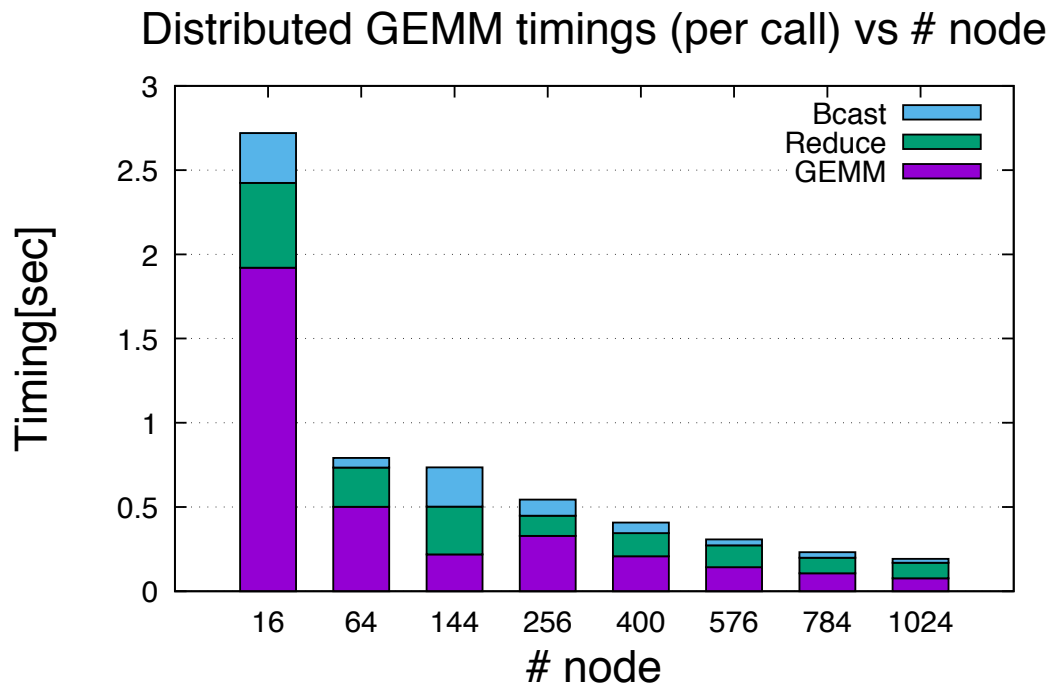

図 4.16　各ノード数における block BiCGrQ 法内部の分散 GEMM の計算時間内訳

表 4.1　計算時間推定に用いた並列 (ノード) 数内訳

P_{int}	1	2	4	4	4	4	4	4	4	4	4	4	4
P_{q}	1	1	1	2	4	8	8	8	8	8	8	8	8
P_{LS}	16	16	16	16	16	16	64	144	256	400	576	784	1024
P_{total}	16	32	64	128	256	512	2048	4608	8192	12800	18432	25088	32768

る．SSM-BB について，$T_{\mathrm{LS}}^{(i,j)}(P_{\mathrm{LS}})$ は数値実験 4-2 で計測した時間から算出した反復あたりの平均計算時間と予め計測しておいた反復回数から計算される．SSM-SL については，数値実験 4-2 で計測した時間を全ての領域・積分点における連立一次方程式求解時間として使用する．$T_{\mathrm{other}}(P_{\mathrm{LS}})$ はあらかじめ，予備実験で計測した値を使用する．

ScaLAPACK の PZHEGVX について，2D block cyclic distribution のブロックサイズを各々 1024 とした上で，数値実験 4-2 で両線形ソルバの性能を計測したノード数と同一のノード数において，実際に範囲内のすべての固有値を計算する時間を計測した．結果を図 4.18 に示す．

ノード数が少ない場合，PZHEGVX が最速である．しかし，数 100 ノード以降，速度向上がほとんど得られない状態になっている．SSM-SL および SSM-BB は 16 ノードにおいて PZHEGVX より 3 倍程度計算時間がかかっているが，1000 ノード付近で PZHEGVX を逆転する．SSM-SL は直接法線形ソルバを使用しているので，Top Layer はリニアにスケールし，Middle Layer もほぼリニアにスケールする．一方，Bottom Layer の並列性は数値実験 4-2 で示されている通り良好ではないことから，それ以上の計算時間削減は厳しいと予測される．SSM-BB は Top Layer および Middle Layer の並列性に関してロードバランスが崩れるためにリニアにスケールせず，Middle Layer の並列性を使い切る 512 ノードでは SSM-SL よりも計算時間は遅くなると予測される．しかしながら，block BiCGrQ 法のスケーラビリティは ScaLAPACK の直接法線形ソルバより優れていることから，より多くのノード数を使用できる場合に SSM-SL の計算時間より速くなると予測される．このため，SSM-BB は大規模メニーコア環境の計算リソースを十分に活用することができる固有値解法であると考えられる．

4.4.4　数値実験 4-4

数値実験 4-4 では内部線形ソルバの収束条件が，Sakurai-Sugiura 法で計算される固有値の精度にどのように影響するか調査する．収束判定条件を $10^{-6}, 10^{-8}$ そして 10^{-10} に設定し，各条件について 64 ノードですべての区間の固有対を計算し，その相対残差ノルムにより評価を行った．結果を図 4.19 に示す．

図 4.19 から，より緩い収束判定条件の場合ほど計算される固有対の精度は悪くなることが確認できる．一方で，相対残差ノルムの分布が区間によって均一になっていないことも見て取れる．区間 3 および区間 4 の内部には多くの固有値が含まれており，特に区間 3 では多くの固有値からなるクラスタが形成されている．LM が区間内の固有値数に対して十分大きくない場合に，計算される固有対の精度が悪化することが文献 [21] に示されている．固有値クラスタが形成されている場合，より大きな L

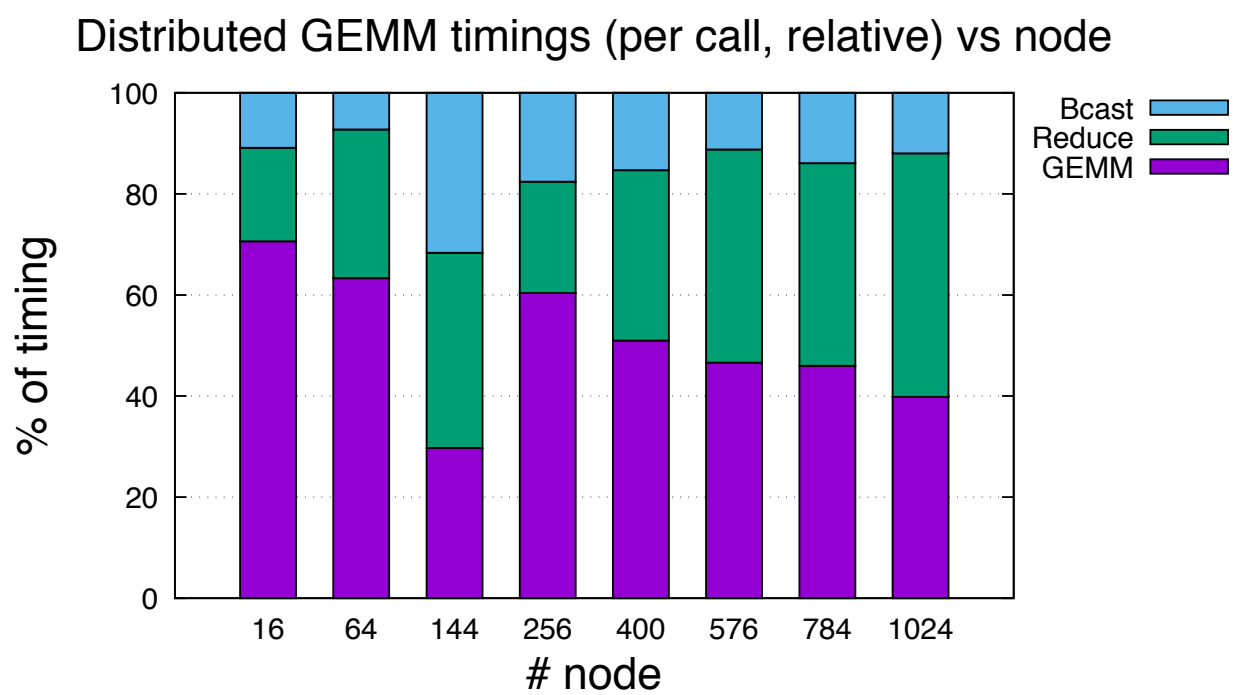

図 4.17　各ノード数における block BiCGrQ 法内部の分散 GEMM の各計算時間の割合

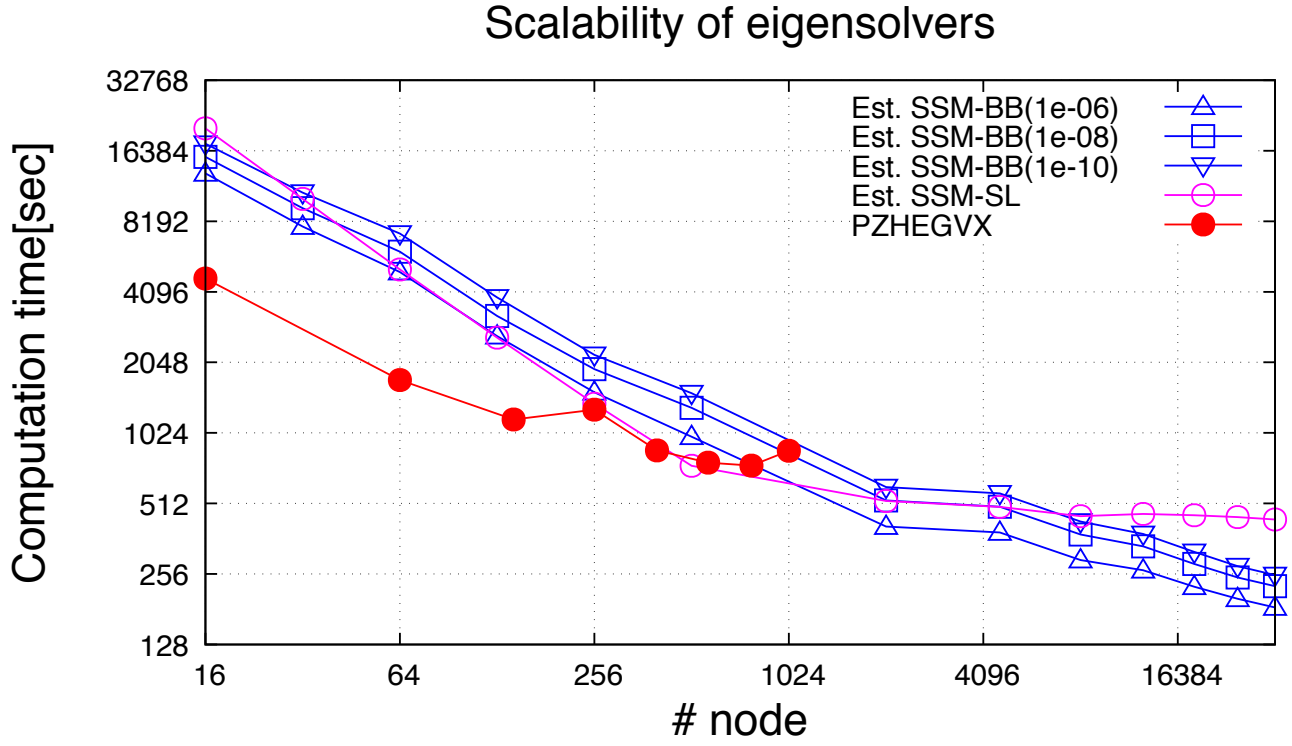

図 4.18　各固有値ソルバの計算時間

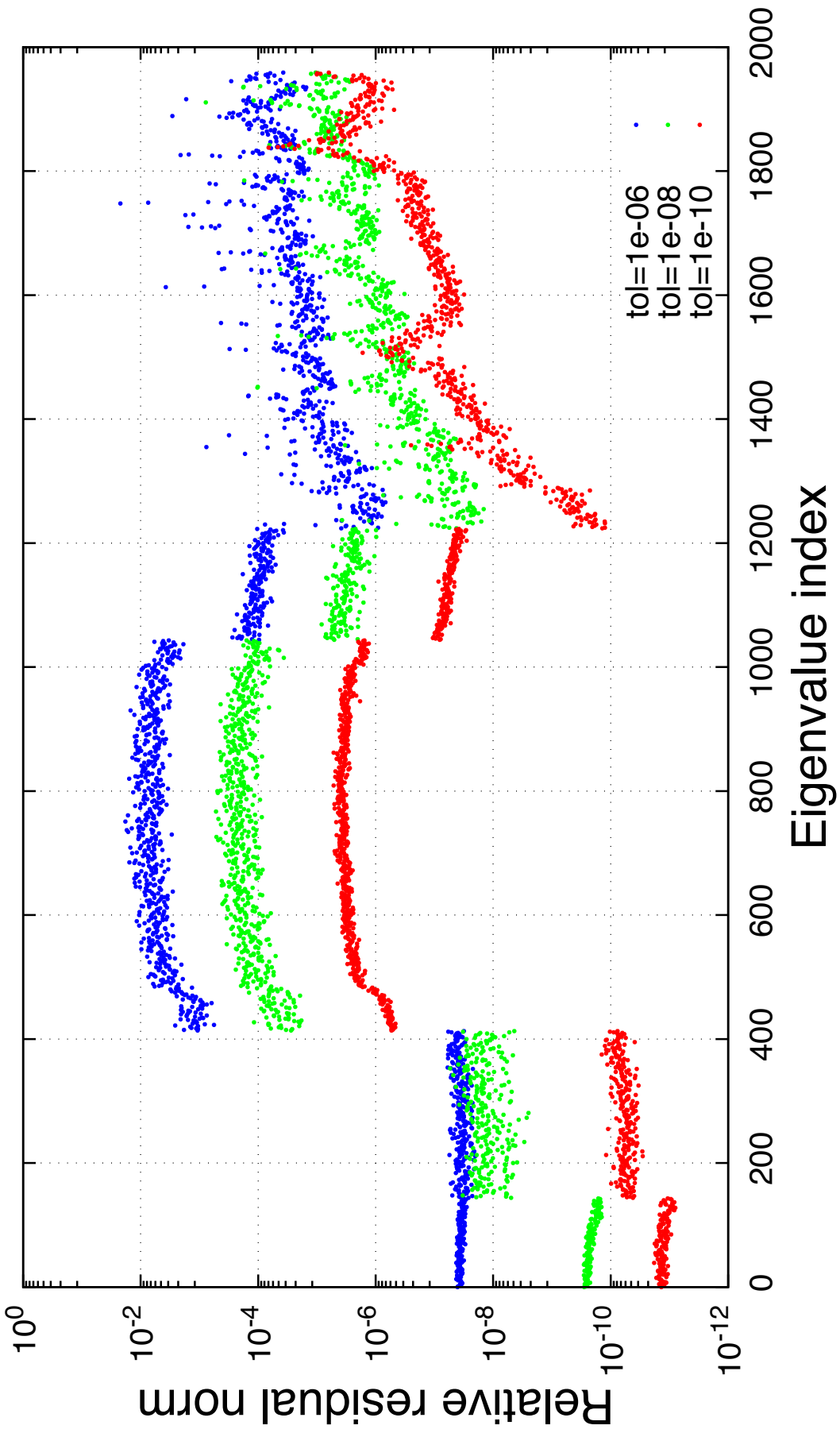

図 4.19　各収束条件における固有対の相対残差ノルム

を用いて計算することで，固有対の精度を改善させることができる．本実験ではすべての区間について同じパラメータを用いて計算を行ったが，すべての区間で同程度の精度を実現したい場合，区間内の固有値分布推定法 [14] など固有値分布に関する情報を用いてパラメータを設定する必要がある．

4.5 小括

本章では，メニーコア計算資源を効率的に活用することができる線形ソルバの分散並列実装として，block Krylov 部分空間反復法を，2D block distribution によるデータ分散手法を採用した分散行列-行列積などを用いて実装する手法を提案した．密行列向けに block Krylov 部分空間反復法を採用すると，直接法に比べて計算量が多くなることから逐次実行においては低速であるが，高並列時には並列化の効きやすさから計算時間が逆転すると考えられ，実問題を用いた数値実験においては実際に計算時間が逆転することを確認した．また，収束判定条件としてより緩い条件を設定した場合に Sakurai-Sugiura 法で計算される固有対の精度がより悪くなることも数値実験により確認された．

第 5 章

複素対称行列向け real valued 解法

前章では，第 3 章で提案した周回積分型固有値解法の分散並列実装を大規模問題に適用するために必要となる，分散並列線形ソルバとして block BiCGrQ 法の分散並列実装を提案した．Block BiCGrQ 法は対象とする問題がエルミート定値一般化固有値問題の場合において，その有用性が示された．

実対称定値一般化固有値問題を周回積分型固有値解法を用いて計算する場合，係数行列が複素対称行列となる連立一次方程式が現れる．この連立一次方程式も block BiCGrQ 法を用いて解くことは可能である．しかし，係数行列や計算過程で出現する行列を全て複素数型で保持する必要があるためメモリ使用量が増大することや，係数行列の対称性を活かすことができないという欠点がある．

本章では，係数行列の対称性を活かすことができ，メモリ使用量を抑えることが可能な新しい block Krylov 部分空間反復法を提案する．

5.1 序説

この章では複数右辺ベクトルを持つ係数行列が複素対称行列な連立一次方程式

$$(A_{\mathrm{R}} + \mathrm{i}A_{\mathrm{I}})(X + \mathrm{i}Y) = B_{\mathrm{R}} + \mathrm{i}B_{\mathrm{I}}, \tag{5.1}$$

を考える．ここで $A_{\mathrm{R}}, A_{\mathrm{I}} \in \mathbb{R}^{n\times n}$ は実対称行列であり，A_{R} は不定値，A_{I} は正定値である．また i は虚数単位であり，$B_{\mathrm{R}}, B_{\mathrm{I}} \in \mathbb{R}^{n\times s}$ は右辺の実部および虚部を表す．$X, Y \in \mathbb{R}^{n\times s}$ は未知行列の実部および虚部である．$s \ll n$ であること，および $(A_{\mathrm{R}} + \mathrm{i}A_{\mathrm{I}})$ は正則であることを仮定する．

複素数で表される問題 (5.1) を等価な 2×2 の実ブロック形式に変換する方法が存在する．例えば，方程式 (5.1) を

$$\begin{pmatrix} A_{\mathrm{R}} & -A_{\mathrm{I}} \\ A_{\mathrm{I}} & A_{\mathrm{R}} \end{pmatrix} \begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} B_{\mathrm{R}} \\ B_{\mathrm{I}} \end{pmatrix} \tag{5.2}$$

と変換することができる．もし実対称な係数行列のものが必要であれば

$$\begin{pmatrix} A_{\mathrm{R}} & A_{\mathrm{I}} \\ A_{\mathrm{I}} & -A_{\mathrm{R}} \end{pmatrix} \begin{pmatrix} X \\ -Y \end{pmatrix} = \begin{pmatrix} B_{\mathrm{R}} \\ B_{\mathrm{I}} \end{pmatrix} \tag{5.3}$$

Algorithm 9 Real-valued 解法の基本アルゴリズム

1: Compute $F = B_{\mathrm{R}} - A_{\mathrm{I}}(A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}(B_{\mathrm{I}} - \gamma B_{\mathrm{R}})$
2: Solve $C_\gamma X = F$
3: Compute $Y = \gamma X - (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}(\gamma B_{\mathrm{R}} - B_{\mathrm{I}} + (1+\gamma^2)A_{\mathrm{I}}X)$

を用いることができる．方程式 (5.2),(5.3) や他の変換方法，およびそれらに対する前処理に関する分析が文献 [8] に記されている．

Axelsson ら [1] は標準的な 2×2 実ブロック形式の Schur complement を基にした real-valued な手法を提案した．その手法は n 次の行列およびベクトルに関する実演算のみ使用している．しかしながら，文献 [1] では A_{R} が正定値であること，および A_{I} が半正定値であることを想定している．これは，この研究で想定している条件と違うため，この方法を直接使用することはできない．文献 [2] ではこの手法のより一般的な非対称行列向けの拡張手法，およびその解析が研究されている．しかし，その手法は行列の対称性や正定値性といった好都合な性質を考慮していない．本章では，A_{R} が不定値，A_{I} が正定値であるような連立一次方程式 (5.1) を解く新しい手法を提案する．提案する手法は文献 [1] の研究からの派生であり，block CG 法と組み合わせることで複数右辺ベクトルを持つ連立一次方程式を効率的に解くことができる．

5.2 Real-valued 解法の定式化

本節では，実演算のみを用いた定式化および基礎となるアルゴリズムについて記述する．基本となる文献 [1] の real-valued アルゴリズムを呼び起こす．方程式 (5.3) は

$$\begin{pmatrix} C_\gamma & 0 \\ \sqrt{1+\gamma^2}A_{\mathrm{I}} & -(A_{\mathrm{R}} + \gamma A_{\mathrm{I}}) \end{pmatrix} \begin{pmatrix} X \\ \frac{1}{\sqrt{1+\gamma^2}}(\gamma X - Y) \end{pmatrix} = \begin{pmatrix} F \\ \frac{\gamma}{\sqrt{1+\gamma^2}}(B_{\mathrm{I}} - \gamma B_{\mathrm{R}}) \end{pmatrix}$$

と書き換えることができる．ここで C_γ および F は，

$$\begin{aligned} C_\gamma &:= A_{\mathrm{R}} - \gamma A_{\mathrm{I}} + (1+\gamma^2)A_{\mathrm{I}}(A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}A_{\mathrm{I}}, \\ F &:= B_{\mathrm{R}} - A_{\mathrm{I}}(A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}(B_{\mathrm{I}} - \gamma B_{\mathrm{R}}), \end{aligned} \tag{5.4}$$

と定義される．$\gamma \in \mathbb{R}$ は $\det(A_{\mathrm{R}} + \gamma A_{\mathrm{I}}) \neq 0$ を満たす値である．詳細については文献 [1] を参照されたい．

上記の関係を利用することで基本となるアルゴリズムを構築することができる．そのアルゴリズムを **Algorithm 9** に示す．大規模疎行列である場合，2 行目の連立一次方程式

$$C_\gamma X = F \tag{5.5}$$

は反復法によって解くことが一般的である．実用上は C_γ は陽に計算せず，block Krylov 部分空間法で必要となる C_γ と $n \times s$ 行列との積を計算する場合には **Algorithm 10** に示す方法にて計算を行う．**Algorithm 9** の 1 行目と 3 行目，そして C_γ の計算の内部で $(A_{\mathrm{R}} + \gamma A_{\mathrm{I}})$ の逆行列を作用させ

Algorithm 10 $W = C_\gamma V$ を計算する手順 $(V, W \in \mathbb{R}^{n\times s})$

1: Compute $T_1 = A_{\mathrm{I}} V$
2: Compute $T_2 = (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1} T_1$
3: Compute $W = A_{\mathrm{R}} V - \gamma T_1 + (1+\gamma^2) A_{\mathrm{I}} T_2$

る計算が現れる．このため，$(A_{\mathrm{R}} + \gamma A_{\mathrm{I}})$ に関する方程式を解く内部線形ソルバが必要となる．ここで，$(A_{\mathrm{R}} + \gamma A_{\mathrm{I}})$ は実対称であるが一般的に不定値である．内部線形ソルバとして，実対称行列向け修正 Cholesky 分解を使用することが考えられる．前処理付き Krylov 部分空間法を用いることもできる．

文献 [1] において，A_{R} が実対称正定値行列および A_{I} が実対称半正定値行列である場合，適切な γ を選択すると $(A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}$ が連立一次方程式 (5.5) を解く際に有効な前処理行列であると説明されている．もし $A_{\mathrm{R}}^{-1} A_{\mathrm{I}}$ の最大固有値が既知である場合，前処理を適用した係数行列の条件数が最小になるような最適な γ を用いることができる．もし最大固有値が既知でない場合においても，$\gamma = 1$ とすれば $\kappa = 2$ となることから，リーズナブルな選択肢として挙げられる．

しかしながら，本研究では A_{R} は不定値であることおよび A_{I} が正定値であることを想定している．そのため，文献 [1] において提案されている好ましい定理が成り立たない．加えて，C_γ および $(A_{\mathrm{R}} + \gamma A_{\mathrm{I}})$ の両者が共に正定値でないことから，実対称正定値行列向けの方法として一番良く知られている (前処理付き)CG 法をそのまま適用することができない．係数行列 C_γ が実対称不定値であることから，短い漸化式を用いた残差ノルムに対して最適となる MINRES 法 [28] を使用することが考えられる．しかしながら，MINRES 法は前処理行列が実対称正定値であることが要求される．実対称正定値な良い前処理行列を探すことは一般的に困難なタスクである．短い漸化式の最適な方法の代わりに，GMRES 法などの長い漸化式を用いた Krylov 部分空間反復法を考えることができる．または，BiCGSTAB 法などの短い漸化式を用いた最適ではない方法の使用も考えられる．

連立一次方程式 (5.5) を解くために短い漸化式の最適な Krylov 部分空間法を使用することができることを示すために，次の命題を示す．

命題 5.2.1. $G_\gamma := (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1} C_\gamma$ を考える．$A_{\mathrm{I}} G_\gamma$ は実対称正定値であり，G_γ は A_{I} 内積に対して対称である．

証明. A_{R} は実対称行列で，A_{I} が実対称正定値行列であることから，$A_{\mathrm{R}} U = A_{\mathrm{I}} U \Lambda$ および $U^{-1} = U^{\mathrm{T}} A_{\mathrm{I}}$ を満たすような正則行列 $U \in \mathbb{R}^{n\times n}$ および対角行列 $\Lambda \in \mathbb{R}^{n\times n}$ が存在する．それらを用いると，G_γ は次のように変形できる．

$$
\begin{aligned}
G_\gamma &= (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1} [A_{\mathrm{R}} - \gamma A_{\mathrm{I}} + (1+\gamma^2) A_{\mathrm{I}} (A_{\mathrm{R}} + \alpha A_{\mathrm{I}})^{-1} A_{\mathrm{I}}] \\
&= [A_{\mathrm{I}} (A_{\mathrm{I}}^{-1} A_{\mathrm{R}} + \gamma I)]^{-1} A_{\mathrm{I}} [A_{\mathrm{I}}^{-1} A_{\mathrm{R}} - \gamma I + (1+\gamma^2)(A_{\mathrm{I}}^{-1} A_{\mathrm{R}} + \gamma I)^{-1}] \\
&= (A_{\mathrm{I}}^{-1} A_{\mathrm{R}} + \gamma I)^{-1} [A_{\mathrm{I}}^{-1} A_{\mathrm{R}} - \gamma I + (1+\gamma^2)(A_{\mathrm{I}}^{-1} A_{\mathrm{R}} + \gamma I)^{-1}] \\
&= U(\Lambda + \gamma I)^{-1} [\Lambda - \gamma I + (1+\gamma^2)(\Lambda + \gamma I)^{-1}] U^{-1} \\
&= U \Theta U^{-1}.
\end{aligned}
$$

ここで，Θ は，$\Theta := (\Lambda+\gamma I)^{-1}[\Lambda-\gamma I+(1+\gamma^2)(\Lambda+\gamma I)^{-1}]$ と定義される．Θ は対角行列であることから，G_γ の固有値 θ は次のように書き表される．

$$\begin{aligned}\theta &= \frac{\lambda-\gamma+(1+\gamma^2)/(\lambda+\gamma)}{(\lambda+\gamma)}\\ &= \frac{1+\lambda^2}{(\lambda+\gamma)^2}.\end{aligned}$$

ここで，λ は Λ の対角要素である．以上から，G_γ の固有値は正の実数である．ただし，G_γ 自身は対称でないことに注意する．次に $A_{\mathrm{I}}G_\gamma$ を考える．$U^{-1}=U^{\mathrm{T}}A_{\mathrm{I}}$ であることから，$A_{\mathrm{I}}G_\gamma = A_{\mathrm{I}}U\Theta(A_{\mathrm{I}}U)^{\mathrm{T}}$ である．この関係から，$A_{\mathrm{I}}G_\gamma$ 実対称正定値行列であり，$(G_\gamma\boldsymbol{x},\boldsymbol{y})_{A_{\mathrm{I}}}=(\boldsymbol{x},G_\gamma\boldsymbol{y})_{A_{\mathrm{I}}}$ となる．ここで，$\boldsymbol{x},\boldsymbol{y}\in\mathbb{R}^n$ であり，$(\boldsymbol{x},\boldsymbol{y})_{A_{\mathrm{I}}}=\boldsymbol{x}^{\mathrm{T}}A_{\mathrm{I}}\boldsymbol{y}$ である．以上により補題 5.2.1 が示された． □

前述の理由により，連立一次方程式 (5.5) と等価な連立一次方程式

$$G_\gamma X=\widehat{F}:=(A_{\mathrm{R}}+\gamma A_{\mathrm{I}})^{-1}F \tag{5.6}$$

を考え，C_γ ではなく $G_\gamma:=(A_{\mathrm{R}}+\gamma A_{\mathrm{I}})^{-1}C_\gamma$ を係数行列とした上で，行列の重み付き内積を使用することで，CG 法や MINRES 法などの最適な Krylov 部分空間法を活用することができるようになる．

5.3 重み付き内積を用いた Krylov 部分空間反復法の構成

本節では，複素対称行列を係数行列とした連立一次方程式 (5.1) を解くための新しい real-valued CG 法を提案する．

非零行列 $V=[\boldsymbol{v}^{(1)},\boldsymbol{v}^{(2)},\ldots,\boldsymbol{v}^{(s)}]$ を用いて，Krylov 部分空間および block Krylov 部分空間は以下の通り定義される．

$$\begin{aligned}\mathcal{K}_m(A;\boldsymbol{v}^{(i)}) &:= \mathrm{span}\{\boldsymbol{v}^{(i)},A\boldsymbol{v}^{(i)},\ldots,A^{m-1}\boldsymbol{v}^{(i)}\},\\ \mathcal{B}_m(A;V) &:= \left\{\sum_{i=1}^{s}\sum_{j=0}^{m-1}\eta_{j,i}A^j\boldsymbol{v}^{(i)} \mid \eta_{j,i}\in\mathbb{C}(\forall j,i)\right\}\\ &= \mathcal{K}_m(A;\boldsymbol{v}^{(1)})+\mathcal{K}_m(A;\boldsymbol{v}^{(2)})+\cdots+\mathcal{K}_m(A;\boldsymbol{v}^{(s)}).\end{aligned}$$

block CG 法 [27] は複数の右辺ベクトルを持つ連立一次方程式向けの CG 法の拡張である．同法は実対称またはエルミート正定値行列を係数行列とする連立一次方程式に対して活用される．

block CG 法の k 番目の反復における，i 番目の右辺ベクトルに対応する残差ベクトル $\boldsymbol{r}_k^{(i)}$ は次の条件を満たす．

$$\boldsymbol{r}_k^{(i)}\perp\mathcal{B}_k(A;R_0).$$

block CG 法では，反復毎に探索空間の基底が s 本づつ拡張される．block CG 法は普通 CG 法より少ない反復回数しか必要としない．

アルゴリズム中の $n \times s$ 行列がほぼ線形従属になりやすいことから，block CG 法を愚直に構築すると不安定になる．そのため，この問題への対処法がいくつか提案されている [27, 26, 11]．

block CG 法の k 反復目の近似解 $\boldsymbol{x}_k^{(i)}$ は全ての $\boldsymbol{x} \in \boldsymbol{x}_0^{(i)} + \mathcal{B}_k(A; R_0)$ に対して

$$\|\boldsymbol{x} - \boldsymbol{x}_i^*\|_A$$

を最小化することが知られている．ここで，$\boldsymbol{x}_i^*$ は i 番目の右辺ベクトルに対する真の解である (文献 [27] の Theorem 2 を見よ)．文献 [27] において，O'Leary は block CG 法の誤差のノルムの上界を示した．

連立一次方程式 (5.6) を解く A_{I} 内積を用いた block CG 法を考える．G_γ は A_{I} 内積に対して対称であることから，以下の条件を満たす block Lanczos 基底 $V_m \in \mathbb{C}^{n \times sm}$ を形成することができる．

$$V_m^{\mathrm{T}} A_{\mathrm{I}} V_m = I,$$
$$V_m^{\mathrm{T}} A_{\mathrm{I}} G_\gamma V_m = T_m.$$

$T_m \in \mathbb{C}^{sm \times sm}$ はブロックサイズが s の対称なブロック三重対角行列である．

そのため，標準の内積を A_{I} 内積に置き換えることで文献 [27] の導出と同様に新しい block CG 法のアルゴリズムは導出される．**Algorithm 11** にアルゴリズムの擬似コードを示す．ここで，C_γ は (5.4) で定義されており，陽には計算しない．C_γ に関する行列積は **Algorithm 10** の手順に従って計算を行う．もし，$(A_{\mathrm{R}} + \gamma A_{\mathrm{I}})$ の $\mathrm{LDL}^{\mathrm{T}}$ 分解が与えられている場合，C_γ に関する積は前進後退代入と行列-行列積のみで計算することが可能である．このアルゴリズムはナイーブな手法であり，5.4 節でより効率的なアルゴリズムを提案する．

文献 [27] において，O'Leary は複数右辺ベクトルを持つ非エルミート行列を係数行列とする連立一次方程式向けの解法として block BiCG 法を提案した．Du らは文献 [10] において，block CG 法に対する Dubrulle[11] の考えを用いて block BiCGrQ 法と呼ばれる安定化した手法を提案した．その block BiCGrQ 法は複素対称行列からなる連立一次方程式向けに特殊化することができ，その特殊化は block COCGrQ 法につながる．

数値実験では提案手法と (前処理付き) block COCGrQ 法との比較を行う．

5.4　効率的な実装

本節では 5.3 節で示した方法の効率的な実装方法を説明する．**Algorithm 11** において，一見すると A_{I} を複数本のベクトル ($n \times s$ 行列) に対して反復あたり 4 回かける必要があるように見える．(2 回は C_γ の計算の内部，残りの 2 回は $\acute{W}_k$ と $\widehat{Z}_{k+1}$ の計算の内部．) 次式の計算で用いられる複数本ベクトルを残しておくことを考える．

$$C_\gamma P_k = A_{\mathrm{R}} P_k - \gamma A_{\mathrm{I}} P_k + (1 + \gamma^2) A_{\mathrm{I}} (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1} A_{\mathrm{I}} P_k.$$

Algorithm 11 複数右辺ベクトルを持つ複素対称行列からなる連立一次方程式を解く real-valued block CG 型解法のナイーブなアルゴリズム

Input: $A_{\mathrm{R}} = A_{\mathrm{R}}^{\mathrm{T}} \in \mathbb{R}^{n\times n}$; $A_{\mathrm{I}} = A_{\mathrm{I}}^{\mathrm{T}} \in \mathbb{R}^{n\times n}$; $B_{\mathrm{R}}, B_{\mathrm{I}}, X_0 \in \mathbb{R}^{n\times s}$; $\gamma \in \mathbb{R}$

Output: $X, Y \in \mathbb{R}^{n\times s}$

1: $F = B_{\mathrm{R}} - A_{\mathrm{I}}(A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}(B_{\mathrm{I}} - \gamma B_{\mathrm{R}})$
2: $\widehat{R}_0 = (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}(F - C_\gamma X_0)$
3: $P_0 = \widehat{R}_0$; $Z_0 = A_{\mathrm{I}}\widehat{R}_0$
4: **for** $k = 0, 1, \ldots$ until solution converges **do**
5: $\quad \acute{S}_k = (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1} C_\gamma P_k$
6: $\quad \acute{W}_k = A_{\mathrm{I}}\acute{S}_k$
7: $\quad \alpha_k = (P_k^{\mathrm{T}}\acute{W}_k)^{-1}(\widehat{R}_k^{\mathrm{T}} Z_k)$
8: $\quad X_{k+1} = X_k + P_k\alpha_k$
9: $\quad \widehat{R}_{k+1} = \widehat{R}_k - \acute{S}_k\alpha_k$
10: $\quad Z_{k+1} = A_{\mathrm{I}}\widehat{R}_{k+1}$
11: $\quad \beta_k = (\widehat{R}_k^{\mathrm{T}} Z_k)^{-1}(\widehat{R}_{k+1}^{\mathrm{T}} Z_{k+1})$
12: $\quad P_{k+1} = \widehat{R}_{k+1} + P_k\beta_k$
13: **end for**
14: $X = X_k$
15: $Y = \gamma X_k - (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}(\gamma B_{\mathrm{R}} - B_{\mathrm{I}} + (1+\gamma^2)A_{\mathrm{I}}X_k)$

ここで以下の通り S_k, T_k そして W_k を定義する．

$$\begin{aligned} S_k &:= A_{\mathrm{I}}P_k, \\ T_k &:= (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1} S_k, \\ W_k &:= C_\gamma P_k. \end{aligned}$$

それらを用いると，

$$\begin{aligned} P_k^{\mathrm{T}}\acute{W}_k &= P_k^{\mathrm{T}} A_{\mathrm{I}}\acute{S}_k = S_k^{\mathrm{T}}\acute{S}_k \\ &= S_k^{\mathrm{T}}(A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1} C_\gamma P_k \\ &= ((A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1} S_k)^{\mathrm{T}} C_\gamma P_k \\ &= T_k^{\mathrm{T}} W_k, \end{aligned}$$

であるから，α_k を $\acute{W}_k = A_{\mathrm{I}}\acute{S}_k$ の計算をせずに計算することができる．加えて，W_k は $W_k = A_{\mathrm{R}}P_k - \gamma S_k + (1+\gamma)^2 A_{\mathrm{I}}T_k$ と計算され，S_k は漸化式 $S_{k+1} = Z_{k+1} + S_k\beta_k$ を用いて更新することができる．このため，A_{I} に関する行列積の回数を 4 回から 2 回に削減することができる．

本来解くべき連立一次方程式は $C_\gamma X = F$ である (**Algorithm 9** を見よ)．残差 $R := F - C_\gamma X$ を手軽に計算するために，漸化式 $R_{k+1} = R_k - W_k\alpha_k$ を導入する．

初期解 X_0 と Y_0 が

$$Y_0 = \gamma X_0 - (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}(\gamma B_{\mathrm{R}} - B_{\mathrm{I}} + (1+\gamma^2)A_{\mathrm{I}}X_0), \tag{5.7}$$

の関係を満たす場合，Y_k は漸化式

$$\begin{aligned} Y_{k+1} &= Y_k + \gamma P_k \alpha_k - (1+\gamma^2)(A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1} A_{\mathrm{I}} P_k \alpha_k \\ &= Y_k + (\gamma P_k - (1+\gamma^2)T_k)\alpha_k, \end{aligned} \tag{5.8}$$

を用いて更新することができる．なぜならば，$X_{k+1} = X_k + P_k\alpha_k$ であり，連立一次方程式 (5.1) の真の解の実部 X^* と虚部 Y^* は

$$Y^* = \gamma X^* - (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}(\gamma B_{\mathrm{R}} - B_{\mathrm{I}} + (1+\gamma^2)A_{\mathrm{I}}X^*)$$

を満たすためである．導出は文献 [1] を参照せよ．

他の Krylov 部分空間法でも見られるように，漸化式で計算した残差 R_k は丸め誤差によって真の残差 $F - C_\gamma X_k$ と異なることがある．そのため，近似解の精度を評価するためにアルゴリズムの最後で真の解を計算する必要がある．

幸いにも真の解は $B_{\mathrm{R}} - A_{\mathrm{R}}X + A_{\mathrm{I}}Y$ で手軽に計算することが可能である．なぜならば，

$$\begin{aligned} R &= F - C_\gamma X \\ &= B_{\mathrm{R}} - A_{\mathrm{I}}(A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}(B_{\mathrm{I}} - \gamma B_{\mathrm{R}}) \\ &\quad - (A_{\mathrm{R}} - \gamma A_{\mathrm{I}} + (1+\gamma^2)A_{\mathrm{I}}(A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}A_{\mathrm{I}})X \\ &= B_{\mathrm{R}} - A_{\mathrm{R}}X + A_{\mathrm{I}}(\gamma X - (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}(\gamma B_{\mathrm{R}} - B_{\mathrm{I}} + (1+\gamma^2)A_{\mathrm{I}}X)) \\ &= B_{\mathrm{R}} - A_{\mathrm{R}}X + A_{\mathrm{I}}Y. \end{aligned} \tag{5.9}$$

この式を用いることで，真の残差の計算のために余計な C_γ の行列積を計算する必要がなくなる．また，この関係を用いることで初期残差 R_0 も少ない計算量で計算することができる．このアイディアは新しいものではない (文献 [1] を参照せよ)．

関係 (5.9) および (5.8) を活用することで，初期化フェーズと最終フェーズ (1,2 行目と 15 行目) における $(A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}$ を作用させる計算を 4 回から 2 回に削減することができる．

他の block Krylov 部分空間反復法と同様に，本手法も $n \times s$ 行列の列ベクトルがほぼ線形従属になりがちであることから，不安定性がある．普通の block CG 法に対して文献 [11] で提案されているように，残差行列 R_k の列ベクトルを thin QR 分解 ($\mathrm{qr}(\cdot)$) を用いることで安定化させることが考えられる．

安定化した手法の構築のために，列直交な行列 $Q_k \in \mathbb{R}^{n\times s}$ を導入する．Q_k は $Q_k\Delta_k = R_k$ を満たす．ただし，$\Delta_k \in \mathbb{R}^{s\times s}$ である．反復中において数値的安定性のために Q_k を R_k の替わりに使用する．R_k を陽に直交化することはしない．Q_k と上三角行列 $\rho_k \in \mathbb{R}^{s\times s}$ は次のように漸化式を用いて計算される．

$$Q_k\rho_k = \mathrm{qr}(Q_{k-1} - \widetilde{W}_{k-1}\widetilde{\alpha}_{k-1}),$$

表 5.1　数値実験で使用した行列の特徴と A_{R} および A_{I} を生成するために使用した複素数スカラー値 z

Name	Size	nnz(A_{R})	nnz(A_{I})	z
BCSST25	15,439	252,241	15,439	100+1i
Ge87H76	112,985	7,892,195	112,985	0+0.01i
VCNT22500	22,500	8,737,290	8,737,290	-0.55+0.01i
CQSZ20	94,948	13,247,276	9,730,976	0.015+4.9e-5i

ここで，$\widetilde{\alpha}_k := \Delta_k \alpha_k \Delta_k^{-1}$ であり，$\widetilde{W_k} := W_k \Delta_k^{-1}$ である．Δ_k は $\Delta_k = \rho_k \Delta_{k-1}$ と更新される．$\|R_k\|_{\mathrm{F}} = \|Q_k \Delta_k\|_{\mathrm{F}} = \|\Delta_k\|_{\mathrm{F}}$ であるから，残差行列のフロベニウスノルムは $\|\Delta_k\|_{\mathrm{F}}$ を経由して計算できることに留意する．

Algorithm 12 にオリジナルの，**Algorithm 13** に残差行列の直交化を行うことで数値的安定性を高めた手法を効率的に計算することができるアルゴリズムを示す．ここで，qr(·) は thin QR 分解を表す．残差直交化を行うバージョンの数式中で現れる α_k, β_k, P_k, S_k, T_k, W_k，そして Z_k はオリジナルの手法の同じ記号で表される量とは数学的に等価ではないことに留意する．本研究では，**Algorithm 13** を提案手法とする．

$\widehat{Q}_k^{\mathrm{T}} Z_k = I_s$ を満たすような $\widehat{Q}_k$ を計算することができることに留意する．ここで，I_s は s 次の単位行列である．この式を用いると，小さな $s \times s$ 行列の逆行列の計算を回避することができる．ρ_k も上三角行列でなければならないわけではないことに留意する．このため，どのような直交化手法でも用いることができる．

5.5　数値実験

本節では提案法の性能を評価するために応用で現れる問題を用いて数値実験を行う．

実対称定値一般化固有値問題 $K\boldsymbol{t} = \mu M\boldsymbol{t}$ を周回積分型固有値解法で解く際に現れる複素対称行列の連立一次方程式に対して提案法を検査する．ここで，K は実対称不定値行列，M は実対称正定値行列であり，$(\mu, \boldsymbol{t})$ は固有対である．周回積分型固有値解法のアルゴリズム中で，複数右辺ベクトルを持つ複素対称行列を係数行列とする連立一次方程式 $(z_j M - K)$ を解く必要がある．ここで，z_j は周回積分を数値積分する際の積分点を示す複素数のスカラー値である．K が不定値，M が正定値であることから，この連立一次方程式は式 (5.1) 型の連立一次方程式である．

表 5.1 に実験で使用した行列の一覧を示す．nnz(·) は行列の非零要素数を表す．BCSST25 は SuiteSparse Matrix Collection[7] より取得した行列対 (BCSSTK25 and BCSSTM25) より生成された問題である．この問題では A_{I} は対角行列である．Ge87H76 も SuiteSparse Matrix Collection[7] より取得した問題である．この問題では A_{I} は単位行列である．VCNT22500 は ELSES matrix library[13] より取得した同名の行列対から生成された問題である．CQSZ20 は密度汎関数理論に基づく電子状態計算アプリケーション CONQUEST [4] で生成された問題である．

実験は 2 基の Intel Xeon E5-2667v3 (2.67GHz, 8 cores/16 thread, Haswells) と 512 GB の主記

表 5.2　数値実験 5-1 の結果 (BCSST25)

s	1	2	4	8	16	32	64
# iter	23	18	14	12	10	8	7
t_{total}	1.0	0.8	1.0	1.4	2.1	3.3	5.8
t_{fact}	0.3	0.2	0.2	0.2	0.2	0.2	0.2
t_{tsol}	0.6	0.7	0.8	1.1	1.7	2.7	4.9
t_{R}	0.0	0.0	0.0	0.0	0.1	0.1	0.1
t_{I}	0.0	0.0	0.0	0.0	0.0	0.0	0.1
t_{qr}	0.0	0.0	0.0	0.0	0.1	0.2	0.3
`true_res`	1.4e-12	1.9e-12	1.9e-12	3.3e-12	3.4e-12	7.1e-12	9.0e-12

憶 (DDR4 ECC REG 32GB×16) が搭載された計算機で行った．アルゴリズムの実装には C++ 言語を用い，線形計算ライブラリ Eigen 3.3.1[12] および Intel Math Kernel Library (MKL) を使用した．実および複素対称 $\mathrm{LDL}^{\mathrm{T}}$ 分解，および全身後退代入は MKL PARDISO を使用した．提案法において，$(A_{\mathrm{R}} + \gamma A_{\mathrm{I}})$ に関する連立一次方程式は PARDISO の直接法ソルバを用いて求解を行った．ILU0 前処理は Eigen の疎行列演算を用いて実装した．

複数右辺ベクトルの実部 B_{R} は乱数により生成した．虚部 B_{I} については，0 とした．反復法の収束条件として $\max_i ||R_k(:,i)||_2/||R_0(:,i)||_2 < 10^{-15}$ を採用した．ここで，$R_k(:,i)$ は R_k の i 番目の列ベクトルを表す．全ての例において初期解は 0 とした．提案法について，A_{R} が正則であることから $\gamma = 0$ とした．

5.5.1　数値実験 5-1

数値実験 5-1 では本章で提案した直交化を行う real-valued block CG 法について，同時に解く右辺ベクトルの数が反復回数や計算時間にどのように影響を及ぼすか調査する．表 5.2,5.3,5.4，そして表 5.5 にそれぞれ，BCSST25, Ge87H76, VCNT22500, そして CQSZ20 に対する結果を示す．これらの表において，# iter は反復回数を表す．t_{fact}, t_{tsol}, t_{R}, t_{I}, そして t_{qr} はそれぞれ実 $\mathrm{LDL}^{\mathrm{T}}$ 分解，前進後退代入, A_{R} および A_{I} に関する行列積，そして QR 分解における計算時間 (単位は秒) を表す．t_{total} は合計の計算時間であり，`true_res` は真の残差ノルムを表す．

実験結果の表から，テストした全ての問題について，右辺ベクトルの数 s が増加すると反復回数が減少していることがわかる．これは，減次された G_γ の最小固有値の増加が収束するために必要な反復回数の削減に寄与していることを示している．

全ての場合において，収束判定条件を満たしているにもかかわらず真の相対残差ノルムは収束判定条件である 10^{-15} より上回っている．これは (block) Krylov 部分空間法においては一般的な事象であり，丸め誤差に起因するものである．全ての場合において，s が増加すると真の残差ノルムも増大していることがわかる．このため，真の残差ノルムに気を払う必要がある．

Algorithm 12 効率化された real-valued block CG 型解法

Input: $A_{\mathrm{R}} = A_{\mathrm{R}}^{\mathrm{T}} \in \mathbb{R}^{n\times n}$; $A_{\mathrm{I}} = A_{\mathrm{I}}^{\mathrm{T}} \in \mathbb{R}^{n\times n}$; $B_{\mathrm{R}}, B_{\mathrm{I}}, X_0, Y_0 \in \mathbb{R}^{n\times s}$; $\gamma \in \mathbb{R}$

Output: $X, Y \in \mathbb{R}^{n\times s}$

if there is no initial guess **then**
  $X_0 = O$; $Y_0 = (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}(B_{\mathrm{I}} - \gamma B_{\mathrm{R}})$
else
  ((5.7) must hold)
end if
$R_0 = B_{\mathrm{R}} - A_{\mathrm{R}}X_0 + A_{\mathrm{I}}Y_0$
$\widehat{R}_0 = (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}R_0$
$P_0 = \widehat{R}_0$; $S_0 = Z_0 = A_{\mathrm{I}}\widehat{R}_0$
for $k = 0, 1, \ldots$ until solution converges **do**
  $T_k = (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}S_k$
  $W_k = A_{\mathrm{R}}P_k - \gamma S_k + (1+\gamma^2)A_{\mathrm{I}}T_k$
  $\alpha_k = (T_k^{\mathrm{T}}W_k)^{-1}(\widehat{R}_k^{\mathrm{T}}Z_k)$
  $X_{k+1} = X_k + P_k\alpha_k$
  $Y_{k+1} = Y_k + (\gamma P_k - (1+\gamma^2)T_k)\alpha_k$
  $R_{k+1} = R_k - W_k\alpha_k$
  $\widehat{R}_{k+1} = (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}R_{k+1}$
  $Z_{k+1} = A_{\mathrm{I}}\widehat{R}_{k+1}$
  $\beta_k = (\widehat{R}_k^{\mathrm{T}}Z_k)^{-1}(\widehat{R}_{k+1}^{\mathrm{T}}Z_{k+1})$
  $P_{k+1} = \widehat{R}_{k+1} + P_k\beta_k$
  $S_{k+1} = Z_{k+1} + S_k\beta_k$
end for
$X = X_k$; $Y = Y_k$

表 5.3　数値実験 5-1 の結果 (`Ge87H76`)

s	1	2	4	8	16	32	64
# iter	37	28	22	16	13	10	8
t_{total}	443.4	467.6	524.9	560.2	639.8	789.5	1038.1
t_{fact}	276.4	282.6	276.3	282.3	276.5	282.1	276.4
t_{tsol}	166.5	184.2	247.3	275.9	359.6	499.8	750.0
t_{R}	0.3	0.5	0.8	1.1	1.8	2.8	4.4
t_{I}	0.0	0.1	0.1	0.1	0.2	0.3	0.5
t_{qr}	0.0	0.1	0.2	0.3	0.7	2.6	3.4
`true_res`	1.2e-14	2.0e-14	3.2e-13	1.1e-12	1.7e-12	3.1e-12	4.3e-12

表 5.4　数値実験 5-1 の結果 (VCNT22500)

s	1	2	4	8	16	32	64
# iter	116	76	48	32	22	17	14
t_{total}	22.7	24.3	26.1	28.8	34.3	50.0	78.9
t_{fact}	1.9	2.0	1.9	2.0	2.0	2.0	1.9
t_{tsol}	17.6	18.2	19.0	19.7	22.5	32.3	51.1
t_{R}	1.0	1.3	1.7	2.2	3.0	4.7	7.7
t_{I}	2.0	2.6	3.2	4.4	5.9	9.2	14.7
t_{qr}	0.0	0.0	0.1	0.1	0.2	0.6	0.8
true_res	7.0e-15	1.5e-14	1.9e-14	1.6e-13	9.2e-13	2.3e-12	4.8e-12

表 5.5　数値実験 5-1 の結果 (CQSZ20)

s	1	2	4	8	16	32	64
# iter	56	38	26	19	14	11	9
t_{total}	101.7	90.6	104.1	108.1	120.1	162.3	232.8
t_{fact}	24.3	23.9	23.9	24.3	23.9	24.3	23.9
t_{tsol}	75.3	63.7	76.2	77.7	86.8	121.3	182.0
t_{R}	0.8	1.0	1.4	2.1	3.1	4.9	7.8
t_{I}	1.2	1.5	2.1	3.0	4.4	6.9	11.1
t_{qr}	0.1	0.1	0.1	0.3	0.6	2.3	3.1
true_res	1.2e-13	1.5e-12	2.5e-12	4.8e-12	5.0e-11	6.4e-11	1.0e-10

5.5.2　数値実験 5-2

数値実験 5-2 では計算時間の観点から複素対称 $\mathrm{LDL}^{\mathrm{T}}$，前処理なし block COCGrQ 法，そして ILU0 前処理付き block COCGrQ 法と比較を行う．表 5.6, 5.7, 5.8，そして表 5.9 にそれぞれ BCSST25, Ge87H76, VCNT22500，そして CQSZ20 に対する結果を示す．

表において，Proposed, C-LDLT, BlockCOCG(I), and BlockCOCG(ILU0) はそれぞれ直交化を行う提案手法，複素対称 $\mathrm{LDL}^{\mathrm{T}}$ 分解を用いた直接法, 前処理を行わない block COCGrQ 法，そして ILU0 前処理付き block COCGrQ 法を表す．

block COCGrQ 法について，複素対称行列に対する行列-ベクトル積に関する計算時間は t_{R} の行に示す．

反復法について，最大反復回数は $\min(\lceil n/s \rceil, 3000)$ と設定した．本数値実験では右辺ベクトル数 s は 16 に固定した．

BCSST25 について，前進後退代入が分解に比べて比較的コストがかかるので，提案法は C-LDLT よ

表 5.6　数値実験 5-2 の結果 (BCSST25)

Method	Proposed	C-LDLT	BlockCOCG(I)	BlockCOCG(ILU0)
# iter	10	–	965(max)	965(max)
t_{total}	2.1	0.6	44.8	68.3
t_{fact}	0.2	0.3	–	0.0
t_{tsol}	1.7	0.2	–	23.9
t_{R}	0.1	–	19.8	20.0
t_{I}	0.0	–	–	–
t_{qr}	0.1	–	16.0	15.8
true_res	3.4e-12	7.9e-13	2.4e+03	8.0e-01

りおよそ 3 倍遅い．BlockCOCG(I) と BlockCOCG(ILU0) については残差ノルムが収束しなかった．

Ge87H76 については，提案法は C-LDLT より高速であった．これは BCSST25 の場合とは対称的に前進後退代入が分解に比べてそこまでコストがかからないことに起因している．BlockCOCG(I) と BlockCOCG(ILU0) の両者は多くの反復回数が必要となり，Proposed と C-LDLT に比べて非常に遅い．ILU0 前処理が反復回数を減らすことに成功しているにもかかわらず，ILU0 の前進後退代入のコストにより合計の計算時間が前処理なしの場合に比べて増大している．加えて，block COCG 法については真の相対残差ノルムが 10^{-11} 付近と収束判定条件の 10^{-15} に比べて非常に大きくなってしまっていることに注意する．

VCNT22500 について，提案法は C-LDLT より 8 倍程度遅くなっている．これは BCSST25 の場合と同様に，前進後退代入のコストが分解に比べて比較的高くなっていることに起因する．BlockCOCG(I) は漸化式で計算した残差のノルムは収束判定条件よりも下回っている．しかし，Proposed と C-LDLT に比べて非常に遅い．BlockCOCG(ILU0) は ILU0 前処理の破綻により計算が終了した．

CQSZ20 について，提案法は C-LDLT よりも低速であった．前進後退代入は $\mathrm{LDL}^{\mathrm{T}}$ 分解に比べて比較的低コストに計算することができたが，反復回数が提案法が C-LDLT より高速になるには十分なほど小さくはならなかった．BlockCOCG(I) と BlockCOCG(ILU0) については残差ノルムが収束しなかった．

図 5.5.2 に各行列に対する $s = 16$ における提案法の残差ノルムの履歴を示す．図から，残差ノルムはスムーズに収束していることがわかる．このような場合，もし低い精度での解を許容できるのであれば，C-LDLT よりも高速になり得る．CQSZ20 はそのような一例である．

最後に，実 $\mathrm{LDL}^{\mathrm{T}}$ 分解を用いた提案法のメモリ要求量は，複素対称 $\mathrm{LDL}^{\mathrm{T}}$ 分解を用いた直接法に比べて半分であることに注意する．大規模な問題では，アルゴリズムのメモリ要求量は計算量と同程度に重要な特徴である．提案法は，メモリ不足が問題となるような問題に対して有用な手法となる．

表 5.7　数値実験 5-2 の結果 (Ge87H76)

Method	Proposed	C-LDLT	BlockCOCG(I)	BlockCOCG(ILU0)
# iter	13	–	2106	1469
t_{total}	639.8	819.1	1759.7	2383.1
t_{fact}	276.5	788.3	–	3.2
t_{tsol}	359.6	30.8	–	1177.5
t_{R}	1.8	–	1212.3	843.6
t_{I}	0.2	–	–	–
t_{qr}	0.7	–	343.0	238.8
true_res	1.7e-12	5.5e-15	4.1e-11	5.6e-11

表 5.8　数値実験 5-2 の結果 (VCNT22500)

Method	Proposed	C-LDLT	BlockCOCG(I)	BlockCOCG(ILU0)
# iter	22	–	1407	Fail
t_{total}	34.3	4.4	860.4	–
t_{fact}	2.0	3.3	–	–
t_{tsol}	22.5	1.1	–	–
t_{R}	3.0	–	807.4	–
t_{I}	5.9	–	–	–
t_{qr}	0.2	–	34.9	–
true_res	9.2e-13	1.4e-15	2.3e-11	–

表 5.9　数値実験 5-2 の結果 (CQSZ20)

Method	Proposed	C-LDLT	BlockCOCG(I)	BlockCOCG(ILU0)
# iter	14	–	3000(max)	3000(max)
t_{total}	120.1	75.4	3404.5	7490.4
t_{fact}	23.9	67.9	–	4.2
t_{tsol}	86.8	7.5	–	4096.4
t_{R}	3.1	–	2774.0	2752.4
t_{I}	4.4	–	–	–
t_{qr}	0.6	–	383.6	391.9
true_res	5.0e-11	2.1e-14	4.5e+01	1.8e+04

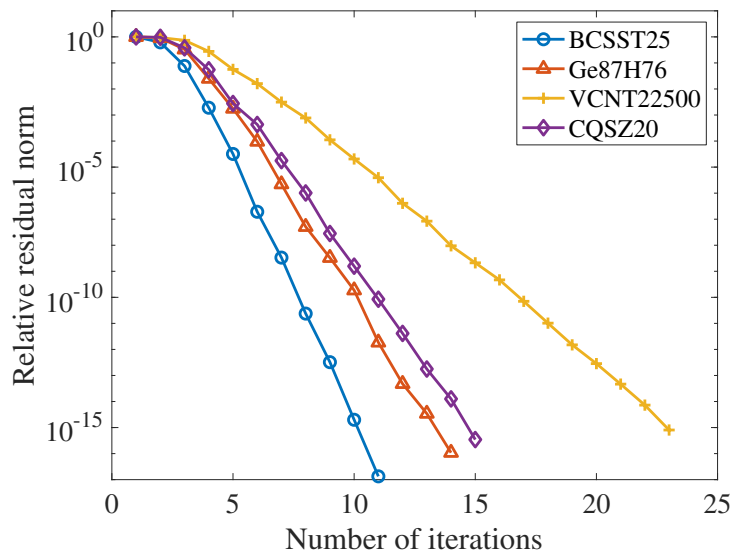

図 5.1　提案法の各行列に対する残差ノルムの履歴

5.6　小括

本章では複数右辺ベクトルを持つ特定の条件の複素対称行列からなる連立一次方程式を解くための新しい block Krylov 部分空間反復法を提案した．係数行列は不定値の実部と正定値の実部を持つことを想定した．

効率的なアルゴリズムの実装手法も提案した．高コストな前進後退代入や虚部の行列積をこの効率的な実装によってその回数を削減することができる．

数値実験において，4 種類のそれぞれ異なる応用で現れる行列に対して右辺ベクトル数を増やした場合に反復回数が減少することを確認した．提案法を複素対称 $\mathrm{LDL}^{\mathrm{T}}$ 分解を用いた直接法，前処理なしそして前処理付き block COCGrQ 法と実験で比較を行った．

結果として，提案法は (ILU0 前処理付き) block COCGrQ 法よりも常に高速になることが判明した．いくつかの場合において，提案法は直接法よりも高速になる場合もあった．しかしながら，前進後退代入の計算コストが比較的高い場合，そして反復回数が十分小さくない場合またはその両方の場合においては直接法よりも高速になることはなかった．

提案法は誤差ノルムを最小化する共役勾配法型の手法である．残差ノルムを最小化する別の最適な real-valued な手法も同様に導出可能である．そのため，今後の課題としてそのような最小残差法に基づく手法を研究することが挙げられる．

Algorithm 13 残差直交化を行う効率化された real-valued block CG 型解法

Input: $A_{\mathrm{R}} = A_{\mathrm{R}}^{\mathrm{T}} \in \mathbb{R}^{n\times n}$; $A_{\mathrm{I}} = A_{\mathrm{I}}^{\mathrm{T}} \in \mathbb{R}^{n\times n}$; $B_{\mathrm{R}}, B_{\mathrm{I}}, X_0, Y_0 \in \mathbb{R}^{n\times s}$; $\gamma \in \mathbb{R}$

Output: $X, Y \in \mathbb{R}^{n\times s}$

if there is no initial guess **then**
  $X_0 = O$; $Y_0 = (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}(B_{\mathrm{I}} - \gamma B_{\mathrm{R}})$
else
  ((5.7) must hold)
end if
$R_0 = B_{\mathrm{R}} - A_{\mathrm{R}}X_0 + A_{\mathrm{I}}Y_0$
$Q_0\Delta_0 = \mathrm{qr}(R_0)$
$\widehat{Q}_0 = (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}Q_0$
$P_0 = \widehat{Q}_0$; $S_0 = Z_0 = A_{\mathrm{I}}\widehat{Q}_0$
for $k = 0, 1, \ldots$ until solution converges **do**
  $T_k = (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}S_k$
  $W_k = A_{\mathrm{R}}P_k - \gamma S_k + (1+\gamma^2)A_{\mathrm{I}}T_k$
  $\alpha_k = (T_k^{\mathrm{T}}W_k)^{-1}(\widehat{Q}_k^{\mathrm{T}}Z_k)$
  $X_{k+1} = X_k + P_k\alpha_k\Delta_k$
  $Y_{k+1} = Y_k + (\gamma P_k - (1+\gamma^2)T_k)\alpha_k\Delta_k$
  $Q_{k+1}\rho_{k+1} = \mathrm{qr}(Q_k - W_k\alpha_k)$
  $\Delta_{k+1} = \rho_{k+1}\Delta_k$
  $\widehat{Q}_{k+1} = (A_{\mathrm{R}} + \gamma A_{\mathrm{I}})^{-1}Q_{k+1}$
  $Z_{k+1} = A_{\mathrm{I}}\widehat{Q}_{k+1}$
  $\beta_k = (\widehat{Q}_k^{\mathrm{T}}Z_k)^{-1}\rho_{k+1}^{\mathrm{T}}(\widehat{Q}_{k+1}^{\mathrm{T}}Z_{k+1})$
  $P_{k+1} = \widehat{Q}_{k+1} + P_k\beta_k$
  $S_{k+1} = Z_{k+1} + S_k\beta_k$
end for
$X = X_k$; $Y = Y_k$

第 6 章

結論

本論文では，大規模計算環境における密一般化固有値問題向けの高性能な固有値解法を実現する手法として，解法として周回積分型固有値解法を採用し，内部の線形ソルバとして線形計算ライブラリ MAGMA の直接法線形ソルバを使用した実装を提案した．ノード内の計算資源だけでは計算できないような大規模な問題に対しても適用できるような線形ソルバとして，大規模密行列向け block Krylov 部分空間反復法を解法として使用することを提案し，その並列実装手法を提案した．また応用上頻出となる実対称定値一般化固有値問題を，周回積分型固有値解法で解く際に現れる複素対称行列を係数行列として持つ連立一次方程式をより効率的に解くことが可能な block Krylov 部分空間反復法を提案した．

第 2 章では，相似変換型解法と周回積分型解法について説明した．

第 3 章では，GPU クラスタにおける中規模密行列向けの分散並列実行可能な固有値解法として，周回積分型固有値解法の MPI および線形計算ライブラリ MAGMA および cuBLAS を使用した実装を提案した．主要な線形計算を GPU を用いて計算を行うことで，CPU のみを用いて計算した場合に比べて大幅な計算速度上昇が期待される．数値実験において，提案した実装は GPU クラスタの計算ノードの性能を十分に発揮することができる固有値解法であることが確認された．一方で，線形ソルバの性能が十分高速になったことにより，直交基底の生成のために行う特異値分解やその後の Rayleigh-Ritz procedure が計算時間のボトルネックになり得ることが示された．

第 4 章では，より大規模な問題を扱うために周回積分型固有値解法の内部で使用する線形ソルバの分散並列化を実現する方法として，block Krylov 部分空間反復法の分散 block GEMM ベースの実装を提案した．block GEMM ベースの実装は，メニーコア環境で高速に計算することが可能な GEMM が主要な計算であること，実装がシンプルであり他の環境への移植および性能のチューニングが容易であること，といった最先端の大規模計算環境に適した性質を保持している．その一方，反復法であることから問題によっては収束しないことや，block Krylov 部分空間反復法においてよく現れる漸化式で計算した残差ノルムと真の残差ノルムの乖離により高精度の解が得られづらいと言った欠点がある．しかしながら，周回積分型固有値解法の内部線形ソルバとして使用することを考えた場合，内部で解く連立一次方程式の厳密ではない解を使用して以後の計算を行った場合には精度の低い固有対が得られることが経験的に知られている．この性質を利用して，もしアプリケーション側が精

度の低い固有対を許容するのであれば，収束判定条件を緩く設定することにより反復回数を削減することで計算量を削減することが可能である．数値実験では Knights Landing 世代の Xeon Phi クラスタ上において，ScaLAPACK の直接法線形ソルバと計算時間および強スケーリングの比較を行った．また，併せて周回積分型固有値解法の内部線形ソルバとして使用した場合に，ScaLAPACK の直接法固有値解法と固有値解法としての計算時間の比較を行った．ScaLAPACK の直接法線形ソルバは OpenMP によってノード内の計算時間がノード間通信が相対的に計算時間の主要部になる程度に削減されているため，ほとんどノード間並列の利点を活かせていない一方で，block GEMM ベースの反復法は GEMM の性能を活かしてより良い強スケーリング性能を達成している．固有値解法としての性能比較では，数 100 ノードで ScaLAPACK の直接法固有値ソルバは性能が頭打ちになる一方で，周回積分型固有値解法と block Krylov 線形ソルバを組み合わせた手法では数万ノード程度まで計算速度が改善する可能性が示されている．このため，同手法は最先端の超並列計算機の性能をフルに活用することができる解法であると考えられる．また，内部線形ソルバの収束判定条件により固有値解法として得られる固有対の精度についても検証を行い，より緩い収束条件を課した場合に，より精度の低い固有値が得られることが示された．以上により，密行列向けの線形ソルバとして反復法を採用した提案法の有用性が示された．

第 5 章では，応用上重要である実対称定値一般化固有値問題を周回積分型固有値解法で解く際に現れる，ある条件を保持する複素対称行列を係数行列として持つ連立一次方程式を効率的に解くための反復法解法として，実演算のみを用いた block Krylov 部分空間反復法を提案した．Axelsson らの先行研究とは条件が違うため，先行研究の定理等をそのまま適用することは不可能であった．しかしながら，行列の重み付き内積を考えることで先行研究の手法を発展させ，短い漸化式で構成される最適な Krylov 部分空間反復法を構築することが可能であることを示した．また，構築した block CG 型解法について，実装の面からより効率的な実装を提案した．数値実験において，提案した real-valued 解法と複素対称 $\mathrm{LDL}^{\mathrm{T}}$ 分解を用いた直接法，そして複素対称行列向け Krylov 部分空間反復法である block COCGrQ 法との比較を複数の種類の行列を使用して行った．提案した real-valued 解法は，右辺ベクトル数を増やすと収束までの反復回数が減少することが実験で使用したすべての行列で示された．他種解法との比較では，提案法は block COCGrQ 法に対しては使用したすべての行列に対して計算時間の面で優位であった．複素対称 $\mathrm{LDL}^{\mathrm{T}}$ 分解を用いた直接法に対しても，特定の条件下においては計算時間で優位になる場合もあることが判明した．メモリ使用量の面では提案法に分があることから，提案法はより大規模な問題に適した解法であると考えられる．

今後の課題として，第 3 章で提案した手法については，最新の Volta 世代の GPU を搭載している環境での実験や他の固有値解法との性能比較を行うことが挙げられる．第 4 章では提案手法を Xeon Phi クラスタにて性能評価を行ったが，GPU クラスタや他のアーキテクチャの計算環境においても性能を発揮することができるか調査を行う必要がある．また，Middle Layer の並列性も実装した上で計算機クラスタの全系レベルを使用した実験を行い，予測した計算時間と比較，検証を行う必要がある．第 5 章で提案した block Krylov 部分空間反復法を第 4 章で提案した分散並列実装手法により実装を行い，周回積分型固有値解法に適用し，計算時間及び消費メモリ量の観点から評価を行う必要がある．

謝辞

本論文の執筆に当たり，常日頃からご指導いただきました筑波大学 システム情報系 櫻井鉄也 教授に心から感謝いたします．博士前期課程在学時に国際会議への論文投稿の機会を与えていただき，またその論文が国際会議の最優秀論文賞として採択されたことは研究生活において大きな自信になりました．博士後期課程進学後は自分の至らなさから度々迷惑をおかけいたしましたが，懇切丁寧に指導していただき，本当にありがとうございました．

大変お忙しい中本論文の審査を引き受けてくださいました，筑波大学 計算科学研究センター 高橋大介 教授，建部修見 教授，小野倫也 准教授，そして多田野寛人 助教に心から感謝すると共に御礼申し上げます．

研究を行う上で快適な環境を提供していただきました，筑波大学 システム情報系 北川高嗣 教授に心から感謝いたします．

研究を行うために必要な行列データを提供していただきました，Swiss National Computing Centre, Thomas C. Schulthess 教授，Anton Kozhevnikov 博士に心から感謝いたします．

本研究では，筑波大学計算科学研究センターの学際共同利用プロジェクトにより，HA-PACS ベースクラスタおよび Oakforest-PACS を利用しました．両システムの運用チームの皆様に感謝いたします．

常日頃より指導していただきました，筑波大学 システム情報系 今倉暁 准教授，二村保徳 助教，保國惠一 助教，Claus Aranha 助教，Ranjith Kumar Bakku 助教，叶秀彩 助教に心より感謝いたします．特に二村保徳 助教には，筆者が研究室に配属されたときから熱心に数値計算の理論および並列計算環境における実装方法をご指導いただきました．また，研究室の先達として様々な相談に乗っていただきました．

日頃より過ごしやすく接していただきました，情報数理研究室の学生に心から感謝いたします．

学群・博士前期課程からお世話になりました，現株式会社フィックスターズの平櫛貴章氏，現株式会社富士通ソフトウェアテクノロジーズの藤井久史氏，現株式会社メガチップスの藤浪健太氏，現ソニーセミコンダクタソリューションズ株式会社の丸山裕士氏，そして現株式会社富士通研究所の田渕晶大氏および津金圭佑氏に感謝いたします．課程修了後も度々相談に乗ってくださいましたこと，重ねて感謝いたします．

最後に，中学校卒業後高専への進学から筑波大への編入学，大学院への進学を理解していただき，経済的・精神的に常に支えてくれた両親に心から感謝いたします．

参考文献

[1] O. Axelsson and A. Kucherov, Real valued iterative methods for solving complex symmetric linear systems. *Numerical Linear Algebra with Applications* **7** (4), 2000, pp. 197–218.

[2] O. Axelsson, M. Neytcheva and B. Ahmad, A comparison of iterative methods to solve complex valued linear algebraic systems. *Numerical Algorithms* **66** (4), 2014, pp. 811–841.

[3] W.-J. Beyn, An integral method for solving nonlinear eigenvalue problems. *Linear Algebra and its Applications* **436** (10), 2012, pp. 3839–3863.

[4] *CONQUEST: Linear Scaling DFT*. `http://www.order-n.org/`.

[5] *cuBLAS — NVIDIA Developer*. `https://developer.nvidia.com/cublas`.

[6] J. J. M. Cuppen, A divide and conquer method for the symmetric tridiagonal eigenproblem. *Numerische Mathematik* **36** (2), 1980, pp. 177–195.

[7] T. Davis, Y. Hu and S. Kolodziej, *SuiteSparse Matrix Collection*. `https://sparse.tamu.edu/`.

[8] D. Day and M. A. Heroux, Solving Complex-Valued Linear Systems via Equivalent Real Formulations. *SIAM Journal on Scientific Computing* **23** (2), 2001, pp. 480–498.

[9] I. S. Dhillon, B. N. Parlett and C. Vömel, The Design and Implementation of the MRRR Algorithm. *ACM Transactions on Mathematical Software* **32** (4), 2006, pp. 533–560.

[10] L. Du, Y. Futamura and T. Sakurai, Block conjugate gradient type methods for the approximation of bilinear form $C^{\mathrm{H}}A^{-1}B$. *Computers and Mathematics with Applications* **66** (12), 2014, pp. 2446–2455.

[11] A. A. Dubrulle, Retooling the method of block conjugate gradients. *Electronic Transactions on Numerical Analysis* **12**, 2001, pp. 216–233.

[12] *Eigen*. `http://eigen.tuxfamily.org/`.

[13] *ELSES matrix library*. `http://www.elses.jp/matrix/`.

[14] Y. Futamura, H. Tadano and T. Sakurai, Parallel stochastic estimation method of eigenvalue distribution. *JSIAM Letters* **2**, 2010, pp. 127–130.

[15] G. H. Golub and C. F. Van Loan, *Matrix Computations*. 4th Edition. Johns Hopkins University Press, Baltimore, 2013.

[16] T. Hoshi and T. Fujiwara, Domain boundary formation in helical multishell gold nanowires. *Journal of Physics: Condensed Matter* **21** (27), 2009, 272201, 7pp.

[17] T. Hoshi, S. Yamamoto, T. Fujiwara, T. Sogabe, S.-L. Zhang, An order- N electronic structure theory with generalized eigenvalue equations and its application to a ten-million-atom system. *Journal of Physics: Condensed Matter* **24** (16), 2012, 165502, 5pp.

[18] T. Ikegami and T. Sakurai, Contour integral eigensolver for non-Hermitian systems: a Rayleigh-Ritz-type approach. *Taiwanese Journal of Mathematics* **14** (3A), 2010, pp. 825–837.

[19] T. Ikegami, T. Sakurai and U. Nagashima, A filter diagonalization for generalized eigenvalue problems based on the Sakurai–Sugiura projection method. *Journal of Computational and Applied Mathematics* **233** (8), 2010, pp. 1927–1936.

[20] A. Imakura, L. Du and T. Sakurai, A block Arnoldi-type contour integral spectral projection method for solving generalized eigenvalue problems. *Applied Mathematics Letters* **32**, 2014, pp. 22–27.

[21] A. Imakura, L. Du and T. Sakurai, Error bounds of Rayleigh–Ritz type contour integral-based eigensolver for solving generalized eigenvalue problems. *Numerical Algorithms* **71** (1), 2016, pp. 103–120.

[22] A. Kozhevnikov, A. G. Eguiluz and T. C. Schulthess, Toward First Principles Electronic Structure Simulations of Excited States and Strong Correlations in Nano- and Materials Science. In: *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC '10)*. IEEE Computer Society, 2010, pp. 1–10.

[23] J. Kurzak, P. Luszczek, M. Faverge, J. Dongarra, Programming the LU Factorization for a Multicore System with Accelerators. In: *High Performance Computing for Computational Science - VECPAR 2012*. Ed. by M. Daydé, O. Marques and K. Nakajima. Springer Berlin Heidelberg, 2013, pp. 28–35.

[24] *MAGMA*. `http://icl.cs.utk.edu/magma/`.

[25] D. Mueller, Householder's method for complex matrices and eigensystems of hermitian matrices. *Numerische Mathematik* **8** (1), 1966, pp. 72–92.

[26] A. A. Nikishin and A. Y. Yeremin, Variable Block CG Algorithms for Solving Large Sparse Symmetric Positive Definite Linear Systems on Parallel Computers, I: General Iterative Scheme. *SIAM Journal on Matrix Analysis and Applications* **16** (4), 1995, pp. 1135–1153.

[27] D. P. O'Leary, The block conjugate gradient algorithm and related methods. *Linear Algebra and its Applications* **29**, 1980, pp. 293–322.

[28] C. C. Paige and M. A. Saunders, Solution of Sparse Indefinite Systems of Linear Equations. *SIAM Journal on Numerical Analysis* **12** (4), 1975, pp. 617–629.

[29] E. Polizzi, Density-matrix-based algorithm for solving eigenvalue problems. *Physical Review B* **79** (11), 2009, pp. 115112-1–115112-6.

[30] T. Sakurai and H. Sugiura, A projection method for generalized eigenvalue problems using numerical integration. *Journal of Computational and Applied Mathematics* **159** (1), 2003, pp. 119–128.

[31] *SIRIUS*. `https://github.com/toxa81/sirius`.

[32] A. Stathopoulos and K. Wu, A Block Orthogonalization Procedure with Constant Synchronization Requirements. *SIAM Journal on Scientific Computing* **23** (6), 2002, pp. 2165–2182.

[33] S. Tomov, J. Dongarra and M. Baboulin, Towards dense linear algebra for hybrid GPU accelerated manycore systems. *Parallel Computing* **36** (5-6), June 2010, pp. 232–240.

[34] J. H. Wilkinson, Calculation of the eigenvalues of a symmetric tridiagonal matrix by the method of bisection. *Numerische Mathematik* **4** (1), 1962, pp. 362–367.

[35] J. H. Wilkinson, Calculation of the eigenvectors of a symmetric tridiagonal matrix by inverse iteration. *Numerische Mathematik* **4** (1), 1962, pp. 368–376.

研究業績

査読付き論文

1. T. Yano, Y. Futamura, A. Imakura and T. Sakurai, Performance evaluation of the Sakurai-Sugiura method with a block Krylov subspace linear solver for large dense Hermitian-definite generalized eigenvalue problems, *JSIAM Letters* **10**, 2018, pp. 77-80.
2. Y. Futamura, T. Yano, A. Imakura and T. Sakurai, A real-valued block conjugate gradient type method for solving complex symmetric linear systems with multiple right-hand sides, *Applications of Mathematics* **62** (4), 2017, pp. 333-355.
3. T. Yano, Y. Futamura and T. Sakurai, Multi-GPU scalable implementation of a contour-integral-based eigensolver for real symmetric dense generalized eigenvalue problems. In: *Proceedings of 8th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC-2013)*. IEEE Computer Society, 2013, pp. 121-127.

査読なし国際会議発表

1. T. Yano, Y. Futamura and T. Sakurai, Distributed Parallel Implementation of the Sakurai-Sugiura Method for Large Dense Generalized Eigenvalue Problems, SIAM Conference on Parallel Processing for Scientific Computing (PP18), Tokyo, Japan, Mar. 7-10, 2018.
2. T. Yano, Y. Futamura and T. Sakurai, Parallel implementation of a complex moment based eigensolver for dense generalized eigenvalue problems on distributed GPU systems, 8th International Workshop on Parallel Matrix Algorithms and Applications (PMAA14), Lugano, Switzerland, Jul. 2-4, 2014.
3. T. Yano, Y. Futamura and T. Sakurai, Parallel Implementation of a Contour-Integral-Based Eigensolver for Dense Generalized Eigenvalue Problems on GPU Clusters, International Workshop on Eigenvalue Problems: Algorithms; Software and Applications, in Petascale Computing (EPASA2014), Tsukuba, Ibaraki, Japan, Mar. 7-9, 2014, (Poster Presentation).
4. T. Yano, Y. Futamura and T. Sakurai, Parallel Implementation of a Contour-Integral-

Based Eigensolver for Dense Generalized Eigenvalue Problems on GPU Clusters, SIAM Conference on Parallel Processing for Scientific Computing (PP14), Portland, Oregon, USA, Feb. 18-21, 2014, (Poster Presentation).

査読なし国内会議発表

1. 矢野 貴大, 二村保徳, 今倉暁, 櫻井鉄也, 反復線形ソルバを用いた大規模密一般化固有値問題向け SS-RR 法の性能評価, 日本応用数理学会 2018 年度 年会, 名古屋大学 東山キャンパス, 2018/9/3-5. ポスター発表
2. 矢野 貴大, 二村保徳, 櫻井鉄也, 密行列一般化シフト線形方程式向け Krylov 部分空間反復法の分散並列実装と性能評価, 2017 年並列/分散/協調処理に関する『秋田』サマー・ワークショップ (SWoPP2017), 秋田アトリオンビル, 2017/7/26-28.
3. 矢野 貴大, 二村保徳, 櫻井鉄也, 密一般化固有値問題向け周回積分型固有値解法における連立一次方程式解法の分散並列実装手法, 第 46 回数値解析シンポジウム (NAS2017), グリーンパーク想い出の森, 2017/6/28-30.
4. 矢野 貴大, 二村保徳, 櫻井鉄也, 密行列に対する周回積分型固有値解法の GPU クラスタにおける性能評価, 第 45 回数値解析シンポジウム (NAS2016), 霧島温泉郷 霧島ホテル, 2016/6/8-10.PC 講演発表
5. 矢野 貴大, 二村保徳, 櫻井鉄也, GPU クラスタにおける周回積分型固有値解法の密行列向け並列実装と性能評価, 2015 年並列/分散/協調処理に関する『別府』サマー・ワークショップ (SWoPP2015), ビーコンプラザ 別府国際コンベンションセンター, 2015/8/4-6.
6. 矢野 貴大, 二村保徳, 櫻井鉄也, 周回積分を用いた密一般化固有値問題解法の GPU ライブラリによる並列実装, 神戸大学大学院システム情報学研究科計算科学専攻 協定講座 第五回協定講座シンポジウム, 神戸大学統合研究拠点 コンベンションホール, 2013/9/30.
7. 矢野 貴大, 二村保徳, 櫻井鉄也, 周回積分を用いた実対称密一般化固有値問題解法の GPU ライブラリによる並列実装, 数値線形代数研究集会 (NLAW2013), 東京理科大学 大子研修センター, 2013/8/28-30.
8. 矢野 貴大, 二村保徳, 櫻井鉄也, 複素対称行列を対象とした Real valued 解法について, 数値解析研究集会 (NAW2012), 板橋区立少年自然の家 八ヶ岳荘, 2012/9/6-8.

受賞

1. 筑波大学 平成 26 年度コンピュータサイエンス専攻長賞, 2015.
2. 3PGCIC-2013 Best Paper Award, 2013.